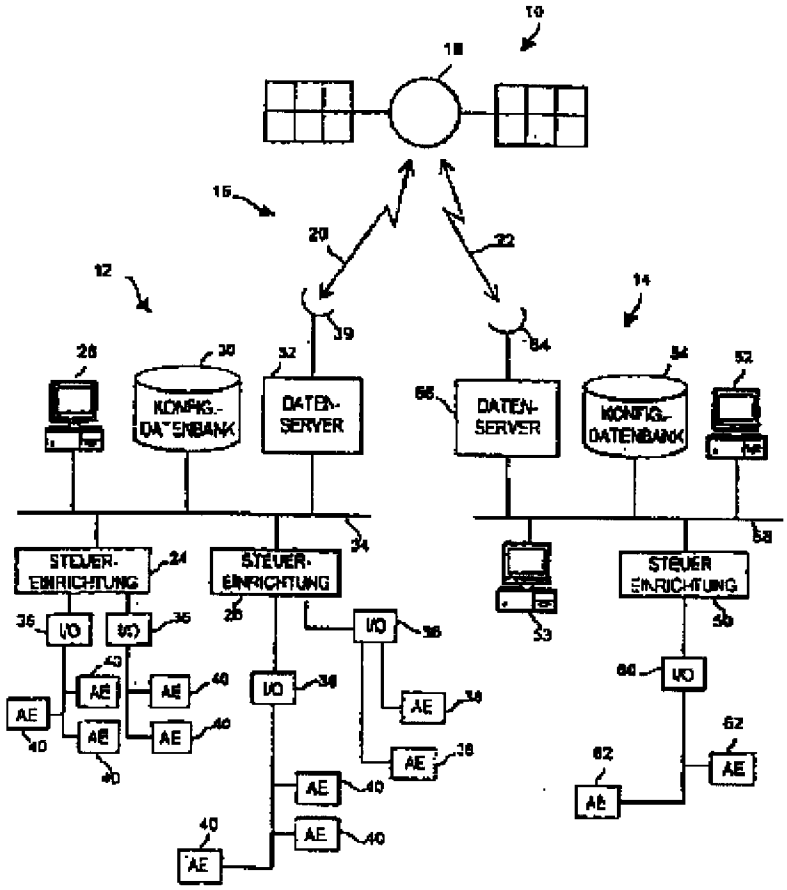
AN: PAT 2001-291834
TI: Configuration data bank system with access and update from separate locations within a process control system e.g. for chemical and crude-oil processing
PN: **DE10049569**-A1
PD: 19.04.2001
AB: NOVELTY - Configuration data banks, as used for the purpose of carrying out a configuration routine for a process control system containing a process control device having at least one host device or operating/servicing work station and several equipment devices, such as valves, valve positioners etc for controlling the operational sequence of the process, are accessed and updated from various locations within the process control system, and are used to store the configuration data which belongs to a particular process control system. A communication network is used to place the number of geographically separate locations in communication with one another and the configuration application is designed so that it communicates with each of the data banks via the communication network and uses data from two or more of the configuration banks so as to carry out a configuration activity, ; USE - Process control systems, such as those in the chemical and crude-oil processing and other processes and especially process control configuration systems provided with a configuration data bank. ADVANTAGE - Enables access and updating of a configuration data bank from different physical locations within a process control system. DESCRIPTION OF DRAWING(S) - A block-diagram of a process control system having two geographically separated locations connected by satellite communication link is given.(Contains non-English language text) . Process control system 10 Satellite link 16 Process control devices 24,26 Network bus 34 Antenna 39 Intelligent equipment devices 40 Process control device 50 User interfaces 52,53 Configuration data bank 54 Network bus 58 I/O device 60 Intelligent equipment devices(64) Antenna 62
PA: (BEOU/) BEOUGHTER K; (CHAT/) CHATKOFF T; (GILB/) GILBERT S; (HAVE/) HAVEKOST R; (NIXO/) NIXON M; (NIXO/) NIXON M J; (ROEC) FISHER-ROSEMOUNT SYSTEMS INC;
IN: CHATKOFF T; GILBERT S; NIXON M; CHATKOFF T A; NIXON M J; BEOUGHTER K; HAVEKOST R;
FA: **DE10049569**-A1 19.04.2001; GB2363872-B 25.08.2004; JP2001184327-A 06.07.2001; GB2363872-A 09.01.2002; US2002111948-A1 15.08.2002; US2003004952-A1 02.01.2003; US6687698-B1 03.02.2004; GB2394816-A 05.05.2004; GB2394816-B 11.08.2004;
CO: DE; GB; JP; US;
IC: G05B-019/042; G05B-019/418; G06F-007/00; G06F-009/445; G06F-012/00; G06F-012/08; G06F-015/16; G06F-015/177; G06F-017/30;
MC: T01-F01B; T01-F05B; T06-A04B1; T06-A04B7; T06-D10;
DC: T01; T06;
FN: 2001291834.gif
PR: US160104P 18.10.1999; US0560199 28.04.2000; US0560361 28.04.2000; US0119269 09.04.2002; US0222555 16.08.2002;
FP: 19.04.2001
UP: 01.09.2004

50


10
18
16
20
22
12
14
39
84
26
30
32
KONFIG.-DATENBANK
DATEN-SERVER
66
DATEN-SERVER
64
KONFIG.-DATENBANK
52
34
58
STEUER-EINRICHTUNG
24
STEUER-EINRICHTUNG
28
STEUER EINRICHTUNG
50
I/O
36
AE
40
38
62
53

(19) BUNDESREPUBLIK DEUTSCHLAND

DEUTSCHES PATENT- UND MARKENAMT

(12) **Offenlegungsschrift**

(10) **DE 100 49 569 A 1**

(51) Int. Cl.[7]:
G 06 F 15/177
G 06 F 9/445
G 05 B 19/418
G 05 B 19/042

(21) Aktenzeichen: 100 49 569.9
(22) Anmeldetag: 6. 10. 2000
(43) Offenlegungstag: 19. 4. 2001

DE 100 49 569 A 1

(30) Unionspriorität:

160104	18. 10. 1999	US
560199	28. 04. 2000	US

(71) Anmelder:
Fisher-Rosemount Systems, Inc., Austin, Tex., US

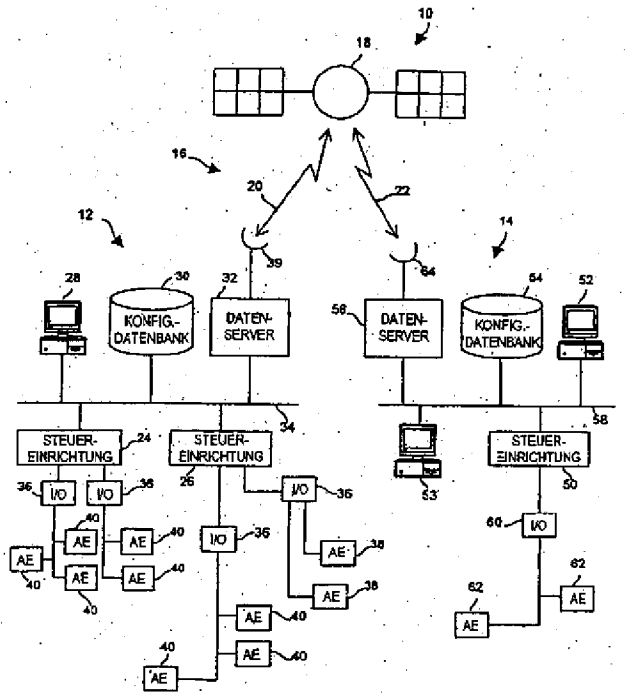
(74) Vertreter:
Meissner, Bolte & Partner, 80638 München

(72) Erfinder:
Nixon, Mark, Round Rock, Tex., US; Gilbert, Stephen, Austin, Tex., US; Chatkoff, Teresa, Austin, Tex., US

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen

(54) Zugriff und Aktualisierung einer Konfigurationsdatenbank von verteilten physischen Orten innerhalb eines Prozeßsteuerungssystems

(57) Ein Konfigurationsdatenbank enthält mehrere Datenbanken, die über eine Vielzahl von physischen Orten innerhalb eines Prozeßsteuersystems verteilt sind. Jede der Datenbanken kann eine Teilmenge der Konfigurationsdaten speichern und diese Teilmenge der Konfigurationsdaten kann durch Benutzer an jedem der Standorte innerhalb des Prozeßsteuersystems zugreifbar sein. Ein Datenbankserver, der einen gemeinsam genutzten Cache hat, greift auf eine Datenbank in einer Weise zu, die es mehreren Teilnehmern ermöglicht, Konfigurationsdaten aus der Datenbank mit einer möglichst geringen Anzahl von Lesevorgängen in der Datenbank zu lesen. Um zu verhindern, daß die Konfigurationsdaten, die von den Teilnehmern in dem Prozeßsteuersystem betrachtet werden, überaltern, erfaßt der Datenbankserver automatisch Veränderungen an einem Element in der Konfigurationsdatenbank und sendet Benachrichtigungen über Veränderunge, die an dem Element vorgenommen wurden, an jeden der Abonnenten dieses Elements, so daß ein Benutzer stets den Status der Konfiguration, der aktuell in der Konfigurationsdatenbank vorliegt, betrachtet.



DE 100 49 569 A 1

Beschreibung

Bei dieser Anmeldung handelt es sich um eine vorschriftsmäßig eingereichte Anmeldung, basierend auf der vorläufigen Anmeldung mit der Serien-Nr. 60/160,104, eingereicht am 18. Oktober 1999 mit dem Titel "Zugriff und Aktualisierung einer Konfigurationsdatenbank von verteilten physischen Orten innerhalb eines Prozeßsteuersystems".

Die vorliegende Erfindung betrifft allgemein Prozeßsteuersysteme und insbesondere ein Prozeßsteuerungskonfigurationssystem, das eine Konfigurationsdatenbank hat, die von geographisch verteilten physischen Orten innerhalb eines Prozeßsteuersystems zugreifbar und aktualisierbar ist.

Prozeßsteuersysteme, wie sie beispielsweise in chemischen und erdölverarbeitenden oder anderen Prozessen verwendet werden, enthalten typischerweise mindestens eine Prozeßsteuereinrichtung, die mit mindestens einer Hosteinrichtung oder Bedienungsworkstation und mit einer oder mehreren Anlageneinrichtungen über analoge und/oder digitale Busleitungen oder andere Kommunikationsleitungen oder -kanäle in Kommunikationsverbindung steht. Die Anlageneinrichtungen, bei denen es sich beispielsweise um Ventile, Ventilpositioniereinrichtungen, Schalter, Messwertgeber (beispielsweise Temperatur-, Druck- und Durchflussmengensensoren) etc. handeln kann, führen Funktionen innerhalb des Prozesses aus, wie beispielsweise das Öffnen oder Schließen von Ventilen und das Messen von Prozeßparametern. Während des Ablaufs eines Prozesses empfängt die Prozeßsteuereinrichtung Signale, welche von den Anlageneinrichtungen durchgeführte Prozeßmessungen und/oder andere Informationen anzeigen, die sich auf die Anlageneinrichtungen beziehen, und zwar über eine Eingabe-/Ausgabe-(I/O)-Einrichtung, verwendet diese Informationen, um eine Steuerroutine zu implementieren und erzeugt anschließend Steuersignale, die über die Busleitungen oder andere Kommunikationskanäle über die Eingabe-/Ausgabeeinrichtung zu den Anlageneinrichtungen gesendet werden, um den Betriebsablauf des Prozesses zu regeln. Informationen von den Anlageneinrichtungen und der Steuereinrichtung werden typischerweise während des Ablaufes einem oder mehreren Anwendungsprogrammen verfügbar gemacht, welche von der Bedienungsworkstation ausgeführt werden, um eine Bedienungsperson in die Lage zu versetzen, jede gewünschte Funktion hinsichtlich des Prozesses auszuführen, beispielsweise den gegenwärtigen Status des Prozesses zu überprüfen. Zusätzlich können Konfigurationsanwendungen, die an einer Benutzerschnittstelle ausgeführt werden, beispielsweise einer Hosteinrichtung, einer Workstation, einem Laptopcomputer etc., verwendet werden, um den Betriebsablauf des Prozesses zu modifizieren, den Prozeß zu konfigurieren, die Konfiguration des Prozesses zu überprüfen, die Prozeßkonfiguration zu dokumentieren etc.

Allgemein werden Prozeßsteuersysteme unter Verwendung einer Konfigurationsdatenbank konfiguriert, in der Konfigurationsinformationen gespeichert sind, die jedem der Hardware- und Softwareelemente im Prozeßsteuersystem zugehörig sind, die Art, in der die Hardwareelemente, wie beispielsweise unterschiedliche Einrichtungen und Steuereinrichtungen in dem Prozeß, physisch miteinander verbunden sind, und die Art und Weise, in der verschiedene Softwareelemente, wie beispielsweise Steuermodule, Kommunikationsmodule etc., den unterschiedlichen Einrichtungen innerhalb des Prozeßsteuersystems zur Durchführung der Prozeßregelung zugehörig sind und von diesen ausgeführt werden. In einigen Fällen ist die Konfigurationsdatenbank eine objektorientierte Datenbank, die Konfigurationsobjekte oder -komponenten für jedes verschiedene logische Element eines Prozeßsteuersystems als Objekte speichert.

Die Konfigurationsdatenbank kann beispielsweise eine Bibliothek enthalten, in der Objektmustervorlagen für einen Teil der oder die gesamten Software- und Hardwareelemente gespeichert sind, welche Mustervorlagen verwendet werden, um Konfigurationsobjekte für Fälle von tatsächlich innerhalb des Prozeßsteuersystems verwendeten Hardware- oder Softwareelementen zu schaffen. Diese Konfigurationsdatenbank kann ferner Setup- oder physische Netzwerkabschnitte enthalten, welche die Art und Weise definieren, in der die physischen Elemente eines Prozeßsteuersystems eingerichtet, verteilt und miteinander verbunden werden. In einigen Fällen enthält die Konfigurationsdatenbank ferner einen Steuerabschnitt, welcher definiert, wie die Steuerung unter Verwendung der Steuereinrichtungen, Anlageneinrichtungen und Steuermodule oder Steuerroutinen, die in den Steuereinrichtungen und/oder den Anlageneinrichtungen durchgeführt werden, ausgeführt wird. Während der Konfiguration des Prozeßsteuersystems wird eine Konfigurationsroutine bzw. -anwendung, die beispielsweise in einer Benutzerschnittstelle oder einer Workstation ausgeführt wird, verwendet, um die Konfigurationsdatenbank zu erstellen oder zu modifizieren, so daß sie die tatsächliche Konfiguration des Prozeßsteuersystems wiedergibt. Diese Konfigurationsanwendung verwendet typischerweise die Informationen innerhalb der Konfigurationsdatenbank, um Einrichtungen, Steuereinrichtungen etc. zu konfigurieren, die zu dem Prozeßsteuersystem gehören, und speichert die neuen Konfigurationsinformationen in der Konfigurationsdatenbank, nachdem eine Konfigurationsaktivität ausgeführt wurde, beispielsweise wenn eine Einrichtung oder ein Softwareelement zu dem System hinzugefügt oder verändert wird etc.

Die Konfigurationsdatenbank wird ferner allgemein verwendet, um die gegenwärtige Konfiguration des Prozeßsteuersystems Benutzern über Benutzerschnittstellen, die mit dem Prozeßsteuersystem verbunden sind, darzustellen. In der Vergangenheit waren einige der Benutzerschnittstellen des Prozeßsteuersystems, die Konfigurationsdatenbanken und die Steuereinrichtungen durch eine eigenständige Busleitung, wie beispielsweise eine Ethernetbusleitung miteinander in Kommunikationsverbindung, so daß ein lokales Netzwerk (LAN) gebildet wurde. Da der eigenständige bzw. dezidierte Ethernetbus eine große Bandbreite hat und da ein bestimmtes Signal oder eine Datenabfrage, die über den Ethernetbus gesendet werden, innerhalb des LAN nicht über sehr große Strecken laufen müssen, ist in diesen Systemen die Kommunikation mit der Konfigurationsdatenbank durch eine der Benutzerschnittstellen sehr direkt und schnell. Als Resultat greifen Konfigurationsdarstellungsroutinen, die in den Benutzerschnittstellen ausgeführt werden, typischerweise auf Konfigurationsinformationen in den Konfigurationsdatenbanken jedesmal dann zu und rufen diese ab, wenn der Benutzer Informationen bezeichnet oder anfordert, die zu der Konfiguration des Prozeßsteuersystems gehören. Diese abgerufenen Informationen werden anschließend über den Ethernetbus ausgesendet und an der Benutzerschnittstelle dem Benutzer dargestellt. Aufgrund der Geschwindigkeit (oder hohen Bandbreite) des dezidierten Ethernetbusses können mehrere Benutzer relativ gleichzeitig auf die Konfigurationsdatenbank zugreifen und dieselben oder unterschiedliche Konfigurationsdaten, die mit der Konfiguration des Prozeßsteuersystems zu tun haben, betrachten. In ähnlicher Weise können verschiedene Benutzer unterschiedliche Teile des Prozeßsteuersystems neu konfigurieren, da alle neuen Konfigurationsdaten, die erzeugt wurden, direkt in relativ kurzer Zeit über den dezidierten Ethernetbus an die Konfigurationsdatenbank abgegeben werden konnten. Da ferner nur diejenigen Einrichtungen, die mit dem LAN ver-

bunden sind, auf die Konfigurationsdatenbank Zugriff haben, und da das LAN typischerweise auf einen einzigen Prozeßort beschränkt ist, besteht kein großer Bedarf, es zu ermöglichen, daß eine große Anzahl von Benutzerschnittstellen auf die Konfigurationsdatenbank gleichzeitig zugreift.

In jüngerer Zeit nimmt jedoch allgemein die Größe einiger Prozeßsteuersysteme zu und es besteht der Wunsch, Konfigurationsinformationen für Prozeßsteuernetzwerke zu integrieren, die über größere bzw. separate geographische Gebiete ausgedehnt oder verteilt sind. Beispielsweise wünscht ein Benutzer die verschiedenen Prozeßstandorten zugehörigen Konfigurationsdaten zu kombinieren, welche in verschiedenen Bezirken, Bundesstaaten oder sogar verschiedenen Ländern befindlich sein können, um es so einer Bedienungsperson an einem ersten Standort zu ermöglichen, auf Konfigurationsinformationen über einen zweiten Standort zuzugreifen und diese zu betrachten und möglicherweise sogar Konfigurationsaktivitäten von dem ersten Standort aus auszuführen, die Auswirkungen auf den zweiten Standort haben. In einem anderen Beispiel kann ein Benutzer den Wunsch haben, eine Ölförderstelle auf einer Bohrplattform viele Meilen vor der Küste mit einem Prozeßsteuersystem einer am Festland befindlichen Ölraffinerie zu integrieren, welches zahlreiche Steuereinrichtungen, Benutzerschnittstellen etc. hat. In diesem Fall ist es wünschenswert, einen Benutzer am Standort der Ölraffinerie in die Lage zu versetzen, die Konfiguration von Einrichtungen auf der Ölförderplattform neu zu konfigurieren oder auf diese einzuwirken, ohne daß es tatsächlich erforderlich ist, zu der Plattform zu fliegen, eine Benutzerschnittstelle an einen Anschluss auf der Ölförderplattform anzuschließen und die Ölfördersteuereinrichtung neu zu konfigurieren, wie es typischerweise gegenwärtig erforderlich ist. Ferner nimmt mit der Integration von mehreren Prozeßsteuerorten die Anzahl von Benutzerschnittstellen, die zum gleichzeitigen Zugriff auf die Konfigurationsdatenbanken dieser Orte verwendet werden können, beträchtlich zu.

Bei der Integration von mehreren geographisch getrennten Orten ist es zwar nicht tatsächlich, jedoch praktisch unmöglich, die verschiedenen Orte unter Verwendung eines gemeinsamen dezidierten Busses, wie z. B. des Ethernetbusses, aufgrund der zu überwindenden Entfernungen zu verbinden. Es ist jedoch möglich, eine Satellitenverbindung, zelluläre Verbindung oder andere Arten der drahtlosen Verbindung oder eine Verbindung über ein Wide Area Network, wie beispielsweise das Internet, oder T1-Leitungen herzustellen, um eine Kommunikationsverbindung zwischen den verschiedenen Orten eines Prozeßsteuersystems zu schaffen und dadurch die Integration von geographisch getrennten Abschnitten eines Prozeßsteuernetzwerkes zu ermöglichen. Die Nutzung von Satellitenverbindungen, zellulären Verbindungen oder anderen drahtlosen Kommunikationsverbindungen über große Distanzen ist typischerweise sehr teuer und unterliegt allgemein gemeinsamer Nutzung, so daß nur eine begrenzte Bandbreite im Vergleich zu einem dezidierten Bus vorhanden ist, wie beispielsweise dem Ethernetbus. Gleichermaßen bieten das Internet, T1-Leitungen und andere gemeinsam genutzte Wide Area Networks nur eine begrenzte Bandbreite bzw. einen begrenzten Durchsatz und sind daher typischerweise für die Datenübertragung im Vergleich zu einem dezidierten Ethernet Local Network Bus sehr langsam. Ferner tragen die geographische Distanz zwischen den unterschiedlichen Orten und die Notwendigkeit, eine sichere und verläßliche Kommunikation zwischen den Orten unter Verwendung von beispielsweise der Bestätigung von Datenpaketen, wie in dem IP-, TCP- und UDP-Kommunikationsprotokoll vorgesehen, zu schaffen, weiter zu der Verzögerung der Kommunikation zwischen den Orten bei.

Als Resultat der Verzögerung der Kommunikation zwischen geographisch voneinander entfernten Standorten eines Prozeßsteuersystems und der Zunahme der Anzahl von Benutzerschnittstellen oder anderen Einrichtungen, die auf eine Konfigurationsdatenbank zugreifen, ist es schwierig, eine Konfigurationsdatenbank zu schaffen, in der die Daten für das gesamte Prozeßsteuersystem, einschließlich der Einrichtungen an jedem der geographisch unterschiedlichen Standorte in einer Weise gespeichert sind, daß sie in derselben Weise von Benutzern an allen unterschiedlichen Standorten zugreifbar sind, sowie in einer Weise, daß sie von Benutzern an den unterschiedlichen Standorten unter Verwendung von gegenwärtigen Konfigurationsdatenbankzugriffsvorgängen geändert werden können. Insbesondere wenn die Kommunikationsdatenbank an einem Hauptstandort angeordnet ist, müssen Benutzer an einem entfernten Standort alle Konfigurationsdaten im Rahmen einer Auffrischung über die langsame bzw. lange Verbindung herunterladen, was eine unangemessen lange Zeitdauer beanspruchen kann. Ferner kann alleine die Erhöhung der Anzahl der Benutzer, die auf die Konfigurationsdatenbank zugreifen, die Zugriffszeit auf jede Information in der Konfigurationsdatenbank auf ein nicht akzeptables Niveau erhöhen. Entsprechend kann bei der Neukonfiguration des Prozeßsteuersystems die Verzögerung der Kommunikation zwischen einem Hauptstandort und einem entfernten Standort es ermöglichen, daß zwei verschiedene Benutzer versuchen, dieselbe Komponente des Prozeßsteuersystems neu zu konfigurieren, was zu Verwirrung bzw. Fehlern führt. So kann beispielsweise ein Benutzer an einem entfernten Standort die gegenwärtige Konfiguration des Prozesses aus der Konfigurationsdatenbank an dem Hauptstandort anfordern und erhalten, einen Teil des Prozeßsteuersystems an dem entfernten Standort neu konfigurieren und anschließend die neuen Konfigurationsdaten zu der Konfigurationsdatenbank an den Hauptstandort senden. In der Zwischenzeit kann jedoch ein Benutzer an dem Hauptstandort denselben Abschnitt des Prozeßsteuersystems neu konfiguriert haben, und kann, da vergleichsweise keine Verzögerung bei der Kommunikation an dem Hauptstandort vorhanden ist, diese neue Konfiguration in der Konfigurationsdatenbank speichern, bevor der Benutzer an dem entfernten Standort versucht, die an dem entfernten Standort erzeugten neuen Konfigurationsdaten zum Abspeichern an die Konfigurationsdatenbank zu senden, was zu Fehlern führt.

In einigen Fällen kann es nicht möglich oder praktisch durchführbar sein, auch nur eine langsame Kommunikationsverbindung bzw. eine Kommunikationsverbindung mit niedriger Bandbreite zwischen unterschiedlichen Standorten zu schaffen, und in diesem Fall muß die Konfiguration eines entfernten Standortes an dem entfernten Standort stattfinden und muß anschließend in die Konfigurationsdatenbank an dem Hauptstandort geladen werden. Diese Off-Line-Konfiguration kann jedoch Probleme verursachen, wenn unterschiedliche Benutzer versuchen, den entfernten Standort zur gleichen Zeit oder nahezu gleichzeitig neu zu konfigurieren, oder wenn ein Benutzer versucht, einen entfernten Standort neu zu konfigurieren, bevor ein anderer Benutzer, der bereits den entfernten Standort neu konfiguriert hat, die Konfigurationsdatenbank aktualisiert, so daß sie die bereits an dem entfernten Standort vorgenommenen Veränderungen wiedergibt.

Es ist Aufgabe der Erfindung, ein Verfahren und eine Vorrichtung zum Zugriff und zur Aktualisierung einer Konfigurationsdatenbank von verschiedenen physischen Orten innerhalb eines Prozeßsteuersystems zu schaffen.

Die Lösung der Aufgabe ergibt sich aus den Patentansprüchen. Unteransprüche beziehen sich auf bevorzugte

Ausführungsformen der Erfindung. Dabei sind auch andere Kombinationen von Merkmalen als in den Ansprüchen beansprucht möglich.

Es wird eine Verteilungs-, Zugriffs- und Änderungsstrategie für eine Konfigurationsdatenbank geschaffen, um es zu ermöglichen, daß zahlreiche geographisch getrennte Orte eines Prozeßsteuersystems, das integriert werden soll, zusammen eine gemeinsame Konfigurationsdatenbank in einer Weise nutzen, die es ermöglicht, daß Benutzer an jedem der verschiedenen Standorte die Konfigurationsdatenbank in zeitlich angemessener, sicherer und konfliktfreier Weise sichten und ändern können, auch wenn die unterschiedlichen Standorte des Prozeßsteuersystems über eine Kommunikationsverbindung mit geringer Bandbreite oder über eine langsame (d. h. verzögerte) Kommunikationsverbindung verbunden sind oder nur mit Unterbrechungen beispielsweise über eine moderne Verbindung in Kommunikationsverbindung stehen.

Gemäß einem Aspekt der Erfindung wird eine Datenbank mit verteilter Konfiguration geschaffen, deren Bestandteile über die verschiedenen geographischen Abschnitte eines Prozeßsteuersystems in der Weise verteilt sind, daß gemeinsam genutzte Konfigurationsdaten, d. h. Konfigurationsdaten, die zu mehr als einem Standort gehören bzw. für mehr als einen Standort anwendbar sind, in einer Datenbank gespeichert sind, die von jedem anderen Standort über eine oder mehrere langsame Kommunikationsverbindungen bzw. Kommunikationsverbindungen mit niedriger Bandbreite zugreifbar ist, und Daten, die nur zu einem bestimmten Standort gehören, in einer Konfigurationsdatenbank an diesem Standort gespeichert sind. Auf diese Weise ist die Konfigurationsinformation, auf die an einem bestimmten Standort höchstwahrscheinlich zugegriffen wird (d. h. die Konfigurationsinformation, die zu diesem Standort gehört), über ein lokales Netzwerk zugreifbar, während Konfigurationsinformationen, die zu anderen Standorten gehört, über eine langsame Kommunikationsverbindung bzw. eine Kommunikationsverbindung mit niedriger Bandbreite zugreifbar sind. Als Resultat müssen nur gemeinsam genutzte Kommunikationsdaten und Daten, die auf einen anderen Standort bezogen sind, über eine langsame Kommunikationsverbindung bzw. eine Kommunikationsverbindung mit niedriger Bandbreite gesendet werden.

Um zu verhindern, daß Konfigurationsdaten, die von Benutzern innerhalb des Prozeßsteuersystems betrachtet werden, überaltert bzw. nicht mehr aktuell werden, sendet das Konfigurationsdatenbankzugriffssystem an jeden Standort gegebenenfalls automatisch Veränderungen, die an der Konfigurationsdatenbank durchgeführt wurden, an jeden der Benutzer, die gegenwärtig diese Daten betrachten, so daß der Benutzer den Status der Konfiguration betrachtet, der tatsächlich in der Konfigurationsdatenbank vorliegt. In einer Ausführungsform enthält jeder Abschnitt der Konfigurationsdatenbank eine Datenzugriffsroutine, die einen gemeinsam genutzten Cache verwendet, um die Ausgabe von Konfigurationskomponenten an jeden der Benutzer oder Teilnehmer, die gegenwärtig diese Daten betrachten, in einer Weise zu koordinieren, welche die Anzahl der Lesevorgänge der Konfigurationsdatenbank reduziert, was die Zugriffsgeschwindigkeit auf Daten von der Konfigurationsdatenbank erhöht. Dieser gemeinsam genutzte Cache kann einen Verriegelungsmechanismus enthalten, um das Verriegeln und Reservieren von Komponenten innerhalb der Konfigurationsdatenbank zu ermöglichen. Nach Wunsch kann die Konfigurationsinformation, die jeder Benutzer erhalten hat, lokal gespeichert werden, um es zu ermöglichen, die Konfigurationsinformation auch dann zu betrachten, wenn die Kommunikationsverbindung zu der Konfigurationsdatenbank, in der die Hauptkopie dieser Information gespeichert ist, unterbrochen wird oder anderweitig nicht zur Verfügung steht.

Nachfolgend wird eine Ausführungsform der Erfindung anhand der Figuren näher erläutert.

Fig. 1 ist ein Blockdiagramm eines Prozeßsteuersystems, das zwei geographisch getrennte Standorte hat, die über eine Satellitenkommunikationsverbindung in Kommunikationsverbindung stehen;

Fig. 2 ist eine Darstellung eines Beispiels einer Prozeßsteuersystemhierarchie, die durch eine Konfigurationsanwendung dargestellt ist;

Fig. 3 ist ein Blockdiagramm eines Prozeßsteuersystems, das einen Hauptstandort und einen entfernten Standort hat, die über eine langsame Kommunikationsverbindung verbunden sind und die eine gemeinsame Master-Konfigurationsdatenbank gemeinsam nutzen;

Fig. 4 ist ein Blockdiagramm eines Prozeßsteuersystems, das einen Hauptstandort und zwei Off-Line-Standorte hat, die alle eine gemeinsame Master-Konfigurationsdatenbank gemeinsam nutzen;

Fig. 5 ist ein Blockdiagramm eines Prozeßsteuersystems, das einen Steuerraum aufweist, der mit einem Anlagenstandort über ein lokales Netzwerk verbunden ist und mit einer Benutzerschnittstelle über eine nicht permanente Kommunikationsverbindung verbunden ist, welche alle eine gemeinsame Master-Konfigurationsdatenbank gemeinsam nutzen;

Fig. 6 ist ein Blockdiagramm eines verteilten Konfigurationsdatenbanksystems, das in einem Prozeßsteuersystem verwendet wird, das verschiedene geographisch getrennte Abschnitte hat, die über langsame Kommunikationsverbindungen bzw. Kommunikationsverbindungen mit geringer Bandbreite in Kommunikationsverbindung stehen;

Fig. 7 ist ein Blockdiagramm eines Client/Server-Systems einer Konfigurationsdatenbank, das einen gemeinsam genutzten Cache innerhalb eines Konfigurationsdatenbankservers nutzt, um eine Subscriber/Publisher-Kommunikation zwischen mehreren Clients und Konfigurationskomponenten, die in einer Konfigurationsdatenbank gespeichert sind, umzusetzen;

Fig. 8 ist ein Objektmodell, das in dem Client/Server-System von **Fig.** 7 verwendet wird, um mehreren Clients Zugriff auf Konfigurationskomponenten zu geben, die in der Konfigurationsdatenbank von **Fig.** 7 gespeichert sind;

Fig. 9 ist ein Blockdiagramm des Zustands der Objekte innerhalb des Datenbankservers von **Fig.** 7 bei Programmbeginn;

Fig. 10 ist ein Blockdiagramm, das die Operation der Objekte in dem Datenbankserver von **Fig.** 7 darstellt, wenn ein erster Client Verbindung mit dem Server aufnimmt, um auf eine erste Konfigurationskomponente von der Konfigurationsdatenbank zuzugreifen;

Fig. 11 ist ein Blockdiagramm, das die Operation der Objekte innerhalb des Datenbankservers von **Fig.** 7 darstellt, wenn der erste Client untergeordnete Elemente einer Komponente von der Konfigurationsdatenbank lädt;

Fig. 12 ist ein Blockdiagramm, das die Operation der Objekte innerhalb des Datenbankservers von **Fig.** 7 darstellt, wenn ein zweiter Client eine Verbindung mit dem Datenbankserver herstellt, um auf eine erste und eine zweite Konfigurationskomponente von der Konfigurationsdatenbank zuzugreifen, die bereits in dem gemeinsam genutzten Cache des Datenbankservers gespeichert sind;

Fig. 13 ist ein Blockdiagramm, das die Operation der Objekte innerhalb des Datenbankservers von **Fig.** 7 ansprechend auf eine Ereignismitteilung darstellt, die durch einen Datenbankserver-Mitteilungsthread erzeugt wurde;

Fig. 14 ist ein Blockdiagramm, das die Operation eines Verriegelungsmanagers in dem Datenbankserver von **Fig.** 7 darstellt, der den gleichzeitigen Zugriff auf den gemeinsam genutzten Cache innerhalb des Datenbankservers vermittelt; und

Fig. 15 ist ein Blockdiagramm, das die Operation der Objekte innerhalb des Datenbankservers von **Fig.** 7 ansprechend auf eine Ereignismitteilung zeigt, die von einem Laufzeitservices-Mitteilungsthread erzeugt wurde.

Wie **Fig.** 1 zeigt, enthält ein Prozeßsteuersystem **10** geographisch voneinander getrennte Orte bzw. Standorte **12** und **14**, die durch eine Satellitenkommunikationsverbindung **16** in Kommunikationsverbindung stehen, die durch einen Satelliten **18** gebildet ist, der einen ersten Kanal **20** mit einer Aufwärts-/Abwärtsstrecke und einen zweiten Kanal **22** mit einer Aufwärts-/Abwärtsstrecke hat, bei welchen es sich um Zweiwegekanäle handeln kann. Der erste Standort **12**, der hier als lokaler oder Hauptstandort **12** bezeichnet wird, enthält zwei Prozeßsteuereinrichtungen **24** und **26**, die mit einer Benutzerschnittstelle **28**, einer Konfigurationsdatenbank **30** und einem Datenserver **32** über einen dezidierten lokalen Netzwerkbus **34** verbunden sind, bei dem es sich beispielsweise um eine Ethernet- oder jede andere gewünschte Bus- oder Kommunikationsleitung handeln kann, verbunden sind. Die Benutzerschnittstelle **28** kann jeder gewünschte Typ einer Host-Workstation oder eines Host-Computers sein, wie beispielsweise jede Art eines Personalcomputers, Laptopcomputers etc., während die Konfigurationsdatenbank **30** eine freistehende Datenbankeinrichtung sein kann oder in eine beliebige andere Einrichtung integriert sein kann, wie z. B. die Benutzerschnittstelle **28** oder den Datenserver **32**. Der Datenserver **32** enthält eine Antenne **39**, mit welcher der Datenserver **32** über die langsame Kommunikationsverbindung **16** mit niedriger Bandbreite mit dem zweiten Prozeßsteuerstandort **14** in Kommunikation steht, der hier als entfernter Standort **14** bezeichnet wird.

Jede der Steuereinrichtungen **24** und **26** ist mit einer oder mehreren Eingabe-/Ausgabeeinrichtungen **36** (I/O) verbunden, welche wiederum mit Anlageneinrichtungen verbunden sind, bei welchen es sich um jede gewünschte Art von Anlageneinrichtungen handeln kann, wie beispielsweise konventionelle 4–20 Milliampere-Einrichtungen **38** oder beliebige intelligente Anlageneinrichtungen **40**, wie beispielsweise HART®-, PROFIBUS®-, Actuator Sensor Interface- (AS-Interface), WORLDFIP®-, Device-Net®-, CAN-, FOUNDATION™- Fieldbus- (nachfolgend als "Fieldbus" bezeichnet) Einrichtungen und dergleichen handeln kann. Selbstverständlich können die Einrichtungen **38** und **40** jede gewünschte Art von Einrichtung darstellen, beispielsweise Sensoren, Sender, Ventile, Gebläse, Mischeinrichtungen, Positioniereinrichtungen etc., die jede Art von Regel-, Meß- oder andere Funktionen innerhalb des Prozeßsteuersystems **10** ausführen können. Die Steuereinrichtungen **24** und **26** können mit den Anlageneinrichtungen **38** und **40** unter Verwendung von beliebigen bekannten oder gewünschten I/O-Einrichtungen **36** und Kommunikationsprotokollen kommunizieren, beispielsweise unter Verwendung der Protokolle, die zu den jeweils vorstehend bezeichneten Arten von Einrichtungen gehören. Allgemein enthalten die Steuereinrichtungen **24** und **26**, bei welchen es sich beispielsweise um DeltaV™-Steuereinrichtungen handeln kann, die von Fisher-Rosemount Systems Inc. vertrieben werden, jeweils einen Prozessor und einen Speicher zum Speichern zum Daten, Programmen und Steuerroutinen (wie z. B. Prozeßsteuermodule), die verwendet werden, um die Steuerung des Prozesses **10** am Hauptstandort **12** umzusetzen. Allgemein ausgedrückt empfangen die Steuereinrichtungen **24** und **26** Signale von den Anlageneinrichtungen **38** und **40**, führen Prozeßsteuerroutinen aus und geben Steuer- bzw. Regelsignale an die Einrichtungen **38** und **40** aus, um dadurch die Regelung des Prozesses **10** umzusetzen.

Entsprechend enthält der entfernte Standort **14** eine Prozeßsteuereinrichtung **50**, die mit Benutzerschnittstellen **52** und **53**, einer Konfigurationsdatenbank **54** und einem Datenserver **56** über einen dezidierten lokalen Netzwerkbus **58** verbunden ist, bei dem es sich beispielsweise um eine Ethernet- oder jede andere gewünschte Bus- oder Kommunikationsleitung handeln kann. Die Steuereinrichtung **50** ist so dargestellt, daß sie durch eine I/O-Einrichtung **60** mit zwei intelligenten Anlageneinrichtungen **62** verbunden ist, sie könnte jedoch auch mit jeder anderen Zahl oder Art von Anlageneinrichtungen verbunden sein. Der Datenserver **56** benutzt eine Antenne **64** zur Kommunikation über eine langsame bzw. mit niedriger Bandbreite arbeitende Datenverbindung **16** mit dem Datenserver **32**, um dadurch die Kommunikation zwischen dem lokalen Standort **12** und dem entfernten Standort **14** sicherzustellen. Die Einrichtungen in dem entfernten Standort **14** können gleich oder ähnlich den entsprechenden Einrichtungen in dem lokalen Standort **12** sein und Prozeßsteuerungs- und Berichtoperationen an dem entfernten Standort **14** durchführen. Dementsprechend versteht sich, daß andere Einrichtungen oder andere Anzahlen von Einrichtungen an einem der beiden oder an beiden Standorten **12** und **14** angeschlossen sein können, um Prozeßsteuerungs- und Konfigurationsaktivitäten in jeder gewünschten Weise durchzuführen. Tatsächlich könnte einer der Standorte, beispielsweise der entfernte Standort **14**, eine einzelne Einrichtung, beispielsweise eine Benutzerschnittstelle, sein, falls dies erwünscht ist. Der Datenserver **32** könnte direkt mit der Konfigurationsdatenbank **30** verbunden sein und in ähnlicher Weise könnte der Datenserver **56** direkt mit der Konfigurationsdatenbank **54** verbunden sein, anstatt daß er durch einen Ethernet-Bus verbunden ist.

Es versteht sich, daß die Datenserver **32** und **56** so arbeiten, daß sie eine Kommunikationsverbindung zwischen den beiden Bussen **34** und **58** herstellen, um dadurch die Kommunikation zwischen den Einrichtungen an dem lokalen Standort **12** und dem entfernten Standort **14** zu ermöglichen. Während die Kommunikationsverbindung **16** als Satellitenverbindung dargestellt ist, kann jede andere beliebige Kommunikationsverbindung anstelle dessen verwendet werden, beispielsweise eine zelluläre Verbindung, eine Modem- oder eine Telefonleitungsverbindung, eine Internetverbindung oder jede andere drahtlose oder Großraumnetzwerk- (WAN) bzw. gemeinsam genutzte lokale Netzwerkverbindung (LAN). Selbstverständlich kann jede gewünschte Kommunikationsstrategie innerhalb der Kommunikationsverbindung **16** verwendet werden, um die Kommunikation von Daten über die Verbindung **16** zu bewerkstelligen. So kann jedes Kommunikationsprotokoll, wie beispielsweise das IP- oder TCP- oder UDP-Protokoll verwendet werden, und jede Modulationstechnik, Fehlercodiertechnik etc. kann verwendet werden, um die Kommunikation über die Verbindung **16** umzusetzen, darunter beispielsweise Spread-Spectrum-Techniken. Verzugsweise wird eine Art von Datenbestätigungsplan in der Kommunikationsverbindung **16** verwendet, um eine sichere und zuverlässige Kommunikation bei Vorhandensein von Rauschen oder anderen Störungen sicherzustellen. Auf Wunsch kann der Datenbestätigungsplan bzw. die Datenbestätigungstechnik, die in der US-Patentanmeldung Seriennummer 09/418,747 mit dem Titel "Deferred Acknowledgment Communications and Alarm Management", eingereicht am 15.10.1999, welche auf den Rechtsinhaber der vorliegenden Erfindung übertragen wurde und die hiermit ausdrücklich durch Bezugnahme hierin eingeschlossen wird, verwendet werden, um die Kommunikation über

die Kommunikationsverbindung **16** zu bewirken. Allgemein ausgedrückt, ermöglicht es die Verwendung der Kommunikationsverbindung **16**, geographisch getrennte Prozeßsteuerstandorte oder -systeme miteinander zu integrieren, um ein einziges Prozeßsteuernetzwerk zu bilden, in dem die Einrichtungen innerhalb eines Standortes mit den Einrichtungen an dem anderen Standort kommunizieren können, um Steuer- und Konfigurationstätigkeiten auszuführen.

Nachteilhafterweise ist die Kommunikation über die Kommunikationsverbindung **16** aufgrund der Distanzen zwischen den Standorten **12** und **14** typischerweise wesentlich langsamer als die Kommunikation über die LAN-Busse **34** und **58**. Ferner kann die Kommunikationsverbindung **16** im Vergleich zu den dezidierten LAN-Bussen **34** und **58** eine niedrige Bandbreite haben und ist allgemein für Rauschen und andere Fehler empfindlicher als die Kommunikation über die LAN-Busse **34** und **58**. Diese Faktoren lassen allgemein die Kommunikationsverbindung **16** zur Ursache von Kommunikationsengpässen innerhalb des Prozeßsteuersystems **10** werden.

Gemäß vorliegender Erfindung sind Konfigurationsdaten, die die Art und Weise betreffen, in der das Prozeßsteuersystem **10** konfiguriert ist, in einer oder in beiden Konfigurationsdatenbanken **30** und **54** gespeichert. Ferner können eine oder mehrere der Benutzerschnittstellen **28**, **52** und **53** oder andere Einrichtungen eine Konfigurationsanwendung speichern und auf Anforderung einer Bedienungsperson diese ausführen. Diese Konfigurationsanwendung kann auf eine oder mehrere der Konfigurationsdatenbanken **30** und **54** zugreifen, um Konfigurationsinformationen zu erhalten, wie beispielsweise Informationen, die die Konfiguration der Einrichtungen, der Softwaremodule etc. innerhalb des Prozeßsteuernetzwerkes **10** betreffen, und kann die erhaltene Konfigurationsinformation auf einem Bildschirm zur Betrachtung durch den Benutzer darstellen. Die Konfigurationsanwendung kann auch einen Benutzer in die Lage versetzen, neue Einrichtungen, Softwareelemente oder andere Elemente zu dem System hinzuzufügen, neue Kommunikationsverbindungen zwischen den Einrichtungen innerhalb des Systems zu schaffen, bereits vorhandene Elemente innerhalb des Systems zu verändern, und dergleichen, um dadurch das Prozeßsteuersystem **10** neu zu konfigurieren. In einem anderen Fall kann eine Konfigurationsanwendung, die in einem ersten Datenbankserver ausgeführt wird, beispielsweise dem Server **32**, auf Informationen von einem anderen Datenbankserver, beispielsweise dem Server **54**, über die langsame Verbindung **16** zugreifen, um diese Informationen einem weiteren, anderen entfernten Standort (in **Fig. 1** nicht dargestellt) zugänglich zu machen. Selbstverständlich kann jede gewünschte Prozeßsteuerkonfigurationsanwendung oder -routine verwendet werden, beispielsweise die Konfigurationsanwendungen, die mit den von Fisher-Rosemount Systems Inc. gelieferten DeltaV-Produkten verbunden sind. Insbesondere die DeltaV-Konfigurationsanwendungen zeigen Prozeßsteuerkonfigurationsinformationen in einem Windowsartigen Explorer Baumformat an und ermöglichen es einem Benutzer oder einer Bedienungsperson, das System durch Drag- und Drop-Verschiebung von Elementen innerhalb des Konfigurationsbaumes auf andere Elemente neu zu konfigurieren, so daß dadurch diese Elemente in unterschiedliche Einrichtungen geladen werden, oder die Plazierung von Prozeßsteuerelementen in verschiedenen Bereichen eines Prozesses anzugeben, die Verbindung von Einrichtungen mit unterschiedlichen Steuereinrichtungen, Bussen oder I/O-Einrichtungen etc. anzugeben, oder Softwareelemente in unterschiedliche Einrichtungen zu laden. Das US-Patent Nr. 5,838,563 für Dove et al. ("System for Configuring a Process Control Environment"), US-Patent-Nr. 5,828,851 für Nixon et al. ("Process Control System Using Standard Protocol Control of Standard Devices and Nonstandard Devices"), die US-Patentanmeldung Nr. 08/631,519 für Nixon et al. ("Process Control System Including a Method and Apparatus for Automatically Sensing the Connection of Devices To a Network"), eingereicht am 12. April 1996, und die US-Patentanmeldung Nr. 08/631,458 für Dove ("System for Assisting Configuring a Process Control Environment"), eingereicht am 12. April 1996, die alle auf den Rechtsinhaber der vorliegenden Erfindung übertragen wurden und alle ausdrücklich durch Bezugnahme hierin eingeschlossen werden, beschreiben Prozeßsteuerkonfigurationsanwendungen, die es einem Benutzer ermöglichen, Prozeßsteuerroutinen und Elemente graphisch zu schaffen, Einrichtungen innerhalb eines Prozeßsteuersystems automatisch zu erfassen und die Steuerung von Einrichtungen innerhalb eines Prozeßsteuersystems zu ermöglichen. Selbstverständlich könnte jede Art von Konfigurationsanwendung in der Benutzerschnittstelle oder in einer Serverdatenbank oder einer anderen Einrichtung ausgeführt werden, um auf die Konfigurationsdatenbanken **30** und **54** zuzugreifen, einschließlich jeder anderen Art von graphischer Konfigurationsanwendung und nicht graphischer Konfigurationsanwendung.

Zur Erläuterung zeigt **Fig. 2** ein Beispiel eines Prozeßsteuerkonfigurationsbaumes **65**, der verwendet werden kann, um eine Konfiguration eines Prozeßsteuersystems darzustellen, und der verwendet werden kann, um Veränderungen an der Konfiguration eines Prozeßsteuersystems graphisch festzulegen. Allgemein ausgedrückt stellt der Konfigurationsbaum **65** aus **Fig. 2** Informationen dar, die zu der Konfiguration einer Betriebsanlage namens Tracey Island gehören. Das Konfigurationssystem von Tracey Island (in dem Baum **65** mit "traceyisland" bezeichnet) enthält einen Bibliotheksabschnitt **66**, der beispielsweise Vorlagen für unterschiedliche Einrichtungskonfigurationen, Einrichtungsdefinitionen, Softwareelemente, wie beispielsweise Steuermodule etc. oder andere Objekte, die in der Anlage Tracey Island verwendet werden, enthalten kann. Obgleich diese Vorlagen nicht in dem Baum **65** dargestellt sind, könnten sie durch Auswählen des Bibliothekssymbols **66** (beispielsweise durch Doppelklicken) dargestellt werden. Der Konfigurationsbaum **65** kann ferner einen Systemkonfigurationsabschnitt **67** aufweisen, der die Art und Weise darstellt, in der die Einrichtungen, wie z. B. Steuereinrichtungen, I/O-Einrichtungen, Anlageneinrichtungen, Benutzerschnittstellen etc. innerhalb des Prozesses physisch angeordnet sind. Die dargestellte Systemkonfiguration für die Betriebsanlage Tracey Island hat Rezepte (welche in der Betriebsanlage beispielsweise zur Herstellung unterschiedlicher Produkte durchzuführende Arbeitsabläufe definieren), Einrichtungsinformationen und Steuerstrategien. Wie dargestellt, hat das Symbol bzw. die Komponente für Steuerstrategien eine Anzahl von untergeordneten Komponenten, darunter nicht zugewiesene I/O-Referenzen, Geräte, und zwei Bereiche (bei denen es sich um physische Bereiche handelt) namens Area_A und Areal. Area_A enthält eine untergeordnete Komponente Module1, die zeigt, daß das Steuermodul Module1 in dem Area_A-Teil der Betriebsanlage Tracey Island geladen ist. Die Systemkonfiguration enthält ferner ein physisches Netzwerk, das eine Liste von außer Betrieb gestellten Steuereinrichtungen hat, sowie ein Steuernetzwerksymbol. Das Steuernetzwerksymbol bzw. die entsprechende Komponente hat einen oder mehrere in Betrieb gestellte Steuereinrichtungen (nur die Steuereinrichtung CTLR1 ist dargestellt) sowie eine Benutzerschnittstelle (mit der Bezeichnung Governmint), die der Operation der Steuereinrichtung CTRL1 zugeordnet ist. Jede Steuereinrichtung

kann eine oder mehrere I/O-Einrichtungen haben, die an dieser angebracht sind, und jede I/O-Einrichtung kann mit einer oder mehreren Anlageneinrichtungen verbunden sein, so daß diese mit den Steuereinrichtungen innerhalb des Baumes **65** in Kommunikationsverbindung sind. Diese Komponenten sind jedoch in **Fig.** 2 nicht dargestellt, könnten jedoch dargestellt werden, wenn beispielsweise das Symbol CTRL1 von einem Benutzer ausgewählt würde.

Es ist zu erkennen, daß die Konfigurationskomponenten, die in **Fig.** 2 dargestellt sind, Parent/Child-Beziehungen haben, wobei beispielsweise die Systemkonfiguration **67** direkt nachfolgende bzw. untergeordnete Elemente, nämlich Rezepte, Setup, Steuerstrategien und physisches Netzwerk, hat. Entsprechend sind die Steuerstrategien ein übergeordnetes Element bzw. ein Parent von nicht zugewiesenen I/O-Referenzen, Geräten, Area_A und Areal. Area_A ist ein Parent der Modul 1-Komponente. Selbstverständlich kann jede der untergeordneten Komponenten weitere untergeordnete Komponenten und so fort haben, so daß der Benutzer jede Ebene des Konfigurationsbaumes **65** betrachten kann, um zu sehen, welche Objekte in welchen Einrichtungen gespeichert sind, welche Einrichtungen in welchen physischen Zonen angeordnet sind, und wie die unterschiedlichen Einrichtungen miteinander in Kommunikationsverbindungen stehen. Allgemein zeigt die Hierarchie von **Fig.** 2 die Anwesenheit von nicht dargestellten untergeordneten Komponenten dadurch an, daß ein Plussymbol (+) neben der übergeordneten bzw. Parent-Komponente plaziert wird. Auch Informationen über eine ausgewählte Komponente einschließlich deren untergeordneter Komponenten können auf der rechten Seite des Bildschirmes dargestellt werden, wie bei der Steuerstrategiekomponente von **Fig.** 2 dargestellt. Selbstverständlich kann weitere Information über diese Komponenten durch Auswählen dieser Komponenten in der Hierarchie von **Fig.** 2 erhalten werden.

Während **Fig.** 2 eine Beispielskonfigurationshierarchie darstellt, versteht es sich, daß die Konfigurationshierarchie für ein Prozeßsteuernetzwerk andere Einrichtungen und Module darstellen kann und andere Abhängigkeiten annehmen kann. Auch kann ein Prozeß in eine Anzahl von physischen Standorten und/oder Zonen unterteilt werden, und jeder dieser Standorte oder jede dieser Zonen kann Steuereinrichtungen, Benutzerschnittstellen etc. haben, die zu dieser gehören, was in der Hierarchie darstellbar ist. Selbstverständlich kann ein Benutzer anfänglich einen Teil des Baumes **65** aus **Fig.** 2 betrachten und anschließend ein Element innerhalb des Baumes **65** auswählen, wie etwa Area_A, um den weiteren Verzweigungen des Konfigurationsbaumes **65** zu folgen. Diese Auswahl veranlaßt, daß die Konfigurationanwendung, die den Hierarchiebaum **65** bereitstellt, auf die untergeordneten Elemente des ausgewählten Objektes zugreift und diese untergeordneten Elemente auf dem Anzeigebildschirm darstellt. In einigen Fällen kann die Konfigurationsinformation als Objekte in einer objektorientierten Datenbank gespeichert werden und die Objekte innerhalb der Datenbank können dieselben Beziehungen wie die in dem Konfigurationsbaum, der von der Konfigurationsanwendung dargestellt wird, gezeigten haben.

Es versteht sich, daß die Konfigurationsobjekte oder -komponenten, die in dem Hierarchiebaum **65** von **Fig.** 2 dargestellt sind, in den Konfigurationsdatenbanken **30** und **54** in **Fig.** 1 gespeichert und in diesen zugreifbar sein können. Die in den Konfigurationsdatenbanken **30** und **54**, bei welchen es sich beispielsweise um objektorientierte Datenbanken jeden gewünschten Typs handeln kann, gespeicherte Information kann in jeder gewünschten Weise gespeichert sein und Beziehungen haben, die den in **Fig.** 2 gezeigten Beziehungen gleich sind oder mit diesen verwandt sind.

Prozeßsteuernetzwerke nach dem Stand der Technik haben typischerweise keine getrennten Standorte, wie etwa Standorte, die über eine langsame oder Langstrecken-Kommunikationsverbindung verbunden sind. Demgemäß enthält jedes Prozeßsteuersystem typischerweise seine eigene Konfigurationsdatenbank, die Konfigurationsdaten hat, die nur die Konfiguration der Einrichtungen innerhalb dieses Systems betreffen. Ferner ist diese Konfigurationsdatenbank typischerweise durch alle Benutzerschnittstellen oder andere Einrichtungen, die mit dem LAN des Prozeßsteuersystems verbunden sind, zugreifbar und kann sehr rasch aufgrund der hohen Bandbreite des LAN-Busses und der relativ geringen Distanz zwischen der Benutzerschnittstelle und der Konfigurationsdatenbank innerhalb des LAN zugegriffen werden. Somit würde beispielsweise bei Systemen nach dem Stand der Technik jeder der Standorte **12** und **14** seine eigene Konfigurationsdatenbank enthalten, die Informationen speichern, welche nur die Einrichtungen innerhalb eines der Standorte, d. h. dem Standort **12** oder dem Standort **14**, jedoch nicht beide betreffen. In diesen Fällen würde zwar die Kommunikationsverbindung **16** zwischen den Standorten **12** und **14** vorgesehen und damit die Kommunikation zwischen diesen Standorten ermöglicht, aber die Konfigurationsdatenbank für einen Standort würde trotzdem nicht die Konfiguration von Einrichtungen innerhalb des anderen Standortes wiedergeben und es gibt keine Möglichkeit, daß eine Konfigurationsanwendung Konfigurationskomponenten, die in getrennten Konfigurationsdatenbanken gespeichert sind, in einer Signalkonfigurationshierarchie oder einem entsprechenden Baum, wie etwa in **Fig.** 2 gezeigt, integriert.

Ferner greifen Konfigurationsanwendungen nach dem Stand der Technik typischerweise jedes Mal dann auf die Konfigurationsdatenbank zu, um die vom Benutzer angeforderte Information zu erhalten, wenn der Benutzer eine tiefergehende Operation oder eine andere Anfrage nach weitergehender Information in der Konfigurationshierachie durchführt. Unglückliderweise sind die tatsächlichen Datenauslesevorgänge aus der Konfigurationsdatenbank der langsamste Teil des Datenabfrageprozesses und somit würden bei einer großen Anzahl von Benutzern, die jeweils die Konfigurationshierarchie betrachten, die Lesevorgänge in der bzw. aus der Konfigurationsdatenbank eine unerwünschte Verzögerung der Möglichkeit, diese Information aus einer einzelnen Datenbank abzurufen, verursachen. Diese Tatsache wird verstärkt, wenn mehrere Benutzer zu dem System über ein oder mehrere entfernte Kommunikationsverbindungen hinzugefügt werden, was die Anzahl der Benutzer erhöht, die in der Lage sind, auf die Konfigurationsdatenbank zuzugreifen. Entsprechend können die Datenabrufvorgänge für diese entfernt angeordneten Benutzer durch die Engstelle der Kommunikation, die die Fernverbindung darstellt, wie etwa die Kommunikationsverbindung **16** in **Fig.** 1, weiter verzögert werden.

Die Datenbank **30** und **54** und die Kommunikation innerhalb des Prozeßsteuersystems **10** können in einer Vielzahl von Arten strukturiert sein, um Konfigurationsinformationen über das Prozeßsteuersystem **10** an zahlreiche Konfigurationsanwendungen entweder nur an dem lokalen Standort **12** oder dem entfernten Standort **14** oder an beiden Standorten abzugeben, jedoch in einer Weise, die das Ausmaß der Kommunikation über die Fernverbindung **16** reduziert, die eine integrierte Ansicht des gesamten oder nur eines Teiles des Prozeßsteuersystems **10** ergibt und die es erlaubt, daß die Konfigurationsinformation von jedem Ort innerhalb eines Prozeßsteuersystems **10** geändert wird. In einem ersten Fall ist eine der Konfigurationsdatenbanken, beispielsweise die Konfigurationsdatenbank in **Fig.** 1, als eine Master-Kon-

figurationsdatenbank bezeichnet, welche das Masterexemplar aller Konfigurationsgegenstände oder Komponenten darin speichert. Die andere Konfigurationsdatenbank **54** wird hierin als eine Aktenkoffer-Datenbank bezeichnet, die auf einen Teil der oder die gesamte Information innerhalb der Master-Konfigurationsdatenbank **30** auf einer Bedarfsbasis zugreift, diese Daten über die Kommunikationsverbindung **16** herunterlädt und Benutzerschnittstellen oder andere Einrichtungen an dem entfernten Standort **14** in die Lage versetzt, diese Daten zu betrachten und zu verändern. Veränderte Daten können über die Kommunikationsverbindung **16** in die Master-Konfigurationsdatenbank **30** hochgeladen werden, um sicherzustellen, daß andere Benutzer über die Master-Konfigurationsdatenbank Zugriff auf die veränderten Daten haben. In diesem Fall werden Aufrufe von einzelnen Benutzerschnittstellen an die Master-Konfigurationsdatenbank **30** über die langsame bzw. mit geringer Bandbreite arbeitende Kommunikationsverbindung **16** reduziert, da alle benötigten Konfigurationsdaten in die Aktenkoffer-Datenbank **54** heruntergeladen werden können und von dieser Datenbank **54** an dem entfernten Standort **14** zugreifbar sind. Ferner ist der Prozeß des Herunterladens einer großen Anzahl von Konfigurationskomponenten in die Aktenkoffer-Datenbank **52** auf einmal hinsichtlich der Nutzung der Kommunikationsverbindung **16** effizienter als der Versuch, diese Information stückweise herunterzuladen.

In einem anderen Fall kann die Konfigurationsdatenbank **54** eine Spiegeldatenbank sein, die den Status der Master-Konfigurationsdatenbank **30** widerspiegelt. Bei der Erstellung kopiert die Spiegeldatenbank **54** alle Daten von der Master-Konfigurationsdatenbank **30**, so daß sie den Status der Datenbank **30** widerspiegelt. Gemäß dieser Strategie greift eine Routine beispielsweise innerhalb der Spiegeldatenbank **54** periodisch auf die Master-Konfigurationsdatenbank **30** über die langsame Kommunikationsverbindung **16** zu und führt eine Synchronisierung mit der Master-Konfigurationsdatenbank **30** durch, indem Veränderungen, die in der Spiegelkonfigurationsdatenbank **54** erfolgt sind, der Master-Konfigurationsdatenbank **30** zur Verfügung gestellt werden, und Veränderungen, die an der Master-Konfigurationsdatenbank **30** durchgeführt wurden, in die Spiegelkonfigurationsdatenbank **54** kopiert werden. Hier greifen die Benutzer an dem Hauptstandort **12** auf die Master-Konfigurationsdatenbank **30** zu, während die Benutzer an dem entfernten Standort **14** auf die Spiegelkonfigurationsdatenbank **54** zugreifen, was die Anzahl der Datenaufrufe, die von einzelnen Benutzern über die langsame Verbindung **16** erfolgen, reduziert bzw. eliminiert.

In einem weiteren Fall sind unterschiedliche Abschnitte der Konfigurationsdaten in den Konfigurationsdatenbanken **30** und **54** gespeichert, so daß eine verteilte Konfigurationsdatenbank geschaffen wird. Eine Konfigurationsanwendung, die in einer Benutzerschnittstelle oder einer anderen Einrichtung ausgeführt wird, greift auf die Konfigurationskomponenten zu, die der Benutzer betrachten oder behandeln möchte, und zwar aus der Datenbank, in der diese Daten ursprünglich gespeichert sind, und abonniert Veränderungen an diesen Komponenten. In einer Ausführungsform werden Abonnentenverbindungen oder -threads zwischen jedem Nutzer oder Client und jeder der Datenbanken, auf die der Benutzer oder Client zugreift, um Konfigurationsdaten abzurufen, geschaffen. Für Konfigurationsdaten, die in der Konfigurationsdatenbank **54** gespeichert sind und die von der Benutzerschnittstelle **53** abonniert sind (beide befinden sich an dem entfernten Standort **14**), erfordert der Thread keine Kommunikation über die Kommunikationsverbindung **16**. Bei Daten, die in der Konfigurationsdatenbank **30** gespeichert sind und von der Benutzerschnittstelle **53** abonniert sind (die an unterschiedlichen Standorten angeordnet sind), erfordert der Thread Kommunikation über die Kommunikationsverbindung **16**. Als Resultat tat es wünschenswert, jedes Stück der Konfigurationsdaten in einer Konfigurationsdatenbank zu speichern, die an dem Standort oder an einer anderen Position angeordnet ist, von der aus auf die Daten am wahrscheinlichsten zugegriffen wird. Die Einzelheiten dieses Schemas werden weiter unten im Detail erläutert.

Die hierin beschriebenen Techniken schaffen einen Weg, daß ein Benutzer Konfigurationsveränderungen an einem entfernten System über eine langsame oder lange Kommunikationsverbindung in Anwesenheit von starkem Rauschen durchführt, was eine Vielzahl von wiederholten Versuchen verursachen kann. Diese Techniken schaffen auch einen Mechanismus, daß Benutzer Teile eines Prozeßsteuersystems konfigurieren können, die von dem Hauptnetzwerk getrennt sind, oder mehrere Prozeßsteuersysteme von derselben Konfigurationseinrichtung aus konfigurieren können. Diese Techniken ermöglichen es auch mehreren Benutzern, an der Konfiguration von verschiedenen Teilen eines Systems aus der Ferne zu arbeiten, was insgesamt die simultane Bearbeitung bei der Entwicklung eines Prozeßsteuersystems verbessern kann.

Das Konzept einer Aktenkoffer-Datenbank zur Verwendung in einem Prozeßsteuersystem wird nachfolgend unter Bezug auf **Fig.** 3 bis 5 beschrieben. Allgemein ausgedrückt ist das Prozeßsteuersystem, das diesen Aufbau verwendet, so konstruiert, daß es eine Master-Konfigurationsdatenbank hat, in der alle Konfigurationsdaten für alle Standorte eines Prozeßsteuersystems gespeichert sind, einschließlich eines Hauptstandortes und verschiedener entfernter Standorte. Der Hauptstandort und die entfernten Standorte können über eine langsame oder mit niedriger Bandbreite arbeitende Kommunikationsverbindung, wie etwa eine Satellitenverbindung oder eine zelluläre Verbindung, in Kommunikationsverbindung stehen, können über ein Modem oder eine Telefonleitungsverbindung mit Unterbrechungen verbunden sein, oder können gar nicht verbunden sein, sondern zum Übertragen von Konfigurationsdaten von einem Standort zum anderen auf einen tragbaren Computer oder eine andere tragbare Speichereinrichtung zurückgreifen. Typischerweise enthält der Hauptstandort die Master-Konfigurationsdatenbank, während jeder entfernte Standort oder ein Laptop oder anderer Computer, der den Hauptstandort mit dem entfernten Standort verbindet, eine Aktenkoffer-Datenbank enthält, die eine Kopie eines Teiles oder der gesamten Konfigurationsdaten in der Master-Konfigurationsdatenbank speichert, die an dem entfernten Standort zu verwenden sind.

Die Aktenkofferkonfigurationsdatenbank wird von der Master-Konfigurationsdatenbank durch einen Benutzer durch eine Herunterlade- und Reservierungsoperation initialisiert. Bei einer Herunterladeoperation kopiert oder erhält die Aktenkoffer-Datenbank heruntergeladene Daten der Abschnitte der Konfigurationsdaten innerhalb der Master-Konfigurationsdatenbank, die erforderlich sind, um an einem entfernten Standort zu arbeiten, diesen neu zu strukturieren, zu verändern, zu verbessern, fehlerfrei zu machen etc. Die Herunterladeoperation kann automatisch die gesamte Konfigurationsinformation herunterladen, die von einem Benutzer benötigt wird, um eine gewünschte Operation an dem entfernten Standort durchzuführen, um den Status des entfernten Standorts zu betrachten etc. Eine Liste aller Konfigurationskomponenten, die für jede Operation erforderlich sind, kann geschaffen werden und gespeichert und benutzt werden, um das Herunterladen durchzuführen. Alternativ kann ein Benutzer die Möglichkeit haben, wichtige Konfi-

gurationsobjekte auszuwählen, die für die Aktivitäten an dem entfernten Standort benötigt werden, und einige oder alle untergeordneten Elemente und/oder übergeordneten Elemente dieser ausgewählten Komponenten können automatisch in die Aktenkoffer-Datenbank heruntergeladen werden. Falls erwünscht, kann jedoch der Benutzer die herunterzuladenden Komponenten auswählen.

Eine Reservierungsoperation wird verwendet, um zu verhindern, daß Konfigurationsdaten innerhalb der Master-Konfigurationsdatenbank geändert werden, während diese Daten von der Aktenkoffer-Datenbank verwendet werden. Nur Elemente, die innerhalb der Master-Konfigurationsdatenbank durch eine Aktenkoffer-Datenbank reserviert wurden, können an dem entfernten Standort geändert werden. Um ein Element zu reservieren, kann eine Konfigurationsanwendung eine Reservierungsmitteilung an die Master-Konfigurationsdatenbank senden, die anschließend dieses Element in der Master-Konfigurationsdatenbank als reserviert markiert und verhindert, daß andere Benutzer Veränderungen an dem Element innerhalb der Master-Konfigurationsdatenbank vornehmen. Das Element kann optional verriegelt werden, wenn es reserviert ist, um Lesevorgänge dieses Elements zu verhindern. Ein derartiger verriegelter Zustand verhindert, daß andere dieses Element editieren oder reservieren. Selbstverständlich können die Herunterlade- und Reservierungsoperation über eine langsame Verbindung, eine Verbindung mit niedriger Bandbreite oder eine nicht permanente Verbindung nach Wunsch durchgeführt werden. Da alle Konfigurationsdaten zusammen gleichzeitig heruntergeladen werden, ist ein derartiges Herunterladen jedoch effizienter als das Übertragen von Kopien der Daten von der Master-Konfigurationsdatenbank über eine langsame oder mit niedriger Bandbreite arbeitende Verbindung auf einzelner Basis nach Bedarf.

Nach dem Herunterladen und Reservieren der Konfigurationselemente wird die Aktenkoffer-Datenbank zu dem entfernten Standort gebracht (sofern die Aktenkoffer-Datenbank nicht bereits an dem entfernten Standort ist) und durch eine Konfigurationsanwendung an dem entfernten Standort benutzt, um Konfigurationsaktivitäten auszuführen. Da die Aktenkoffer-Datenbank mit der gesamten Konfigurationsinformation versehen ist, die erforderlich ist, um eine bestimmte Operation durchzuführen, kann die Aktenkoffer-Datenbank direkt an dem entfernten Standort verwendet werden oder kann verwendet werden, um eine lokale Datenbank an dem entfernten Standort zu schaffen, und diese lokale Datenbank kann von jeder Konfigurationsanwendung genutzt werden, die an dem entfernten Standort ausgeführt wird.

Typischerweise können Konfigurationsanwendungen, die die Aktenkoffer-Datenbank verwenden, nur Elemente editieren, die von der Master-Konfigurationsdatenbank durch diese Aktenkoffer-Datenbank reserviert wurden. Wenn beispielsweise ein Benutzer ein Modul mit einer damit verbundenen zusammengesetzten Datenstruktur reserviert hat, überträgt die Master-Konfigurationsdatenbank das Modul plus alle abhängigen Elemente oder untergeordneten Elemente dieses Moduls in die Aktenkoffer-Datenbank und reserviert nur das Modul. Selbstverständlich können die damit verbundenen zusammengesetzten Datenstrukturen in derselben Weise wie das Modul reserviert werden. Wenn ein Benutzer Herunterladefähigkeit benötigt, erlaubt das System den Benutzern, Herunterladerechte an bestimmten Knoten zu reservieren und Konfigurationsinformationen, einschließlich Herunterladeberichten und Statusinformationen für diese Knoten, mit der Aktenkoffer-Datenbank auszutauschen. Auf Wunsch kann die Konfigurationsanwendung, welche die Aktenkoffer-Datenbank verwendet, in der Aktenkoffer-Datenbank gespeicherte Konfigurationsdaten in einer Weise darstellen, welche diejenigen Elemente deutlich bezeichnet, welche der Benutzer editieren kann. Beispielsweise können editierbare Elemente in dem Konfigurationshierarchiebaum in durchgehender Schrift gekennzeichnet sein, und alle Elemente, die der Benutzer nicht verändern kann, können als grau schattiert dargestellt werden. Selbstverständlich können auch andere Arten der Darstellung verwendet werden, welche Elemente oder Komponenten reserviert wurden. Wie vorstehend angeführt, sind Master-Konfigurationsdatenbankelemente, die reserviert wurden, in der Master-Konfigurationsdatenbank verriegelt, so daß keine Änderungen an diesen vorgenommen werden können, abgesehen von einer Überführungsoperation (nachfolgend beschrieben), die von der Aktenkoffer-Datenbank erzeugt wird, welche diese Elemente reserviert hat. Nachdem der Benutzer Veränderungen an den reservierten Elementen innerhalb der Aktenkoffer-Datenbank vorgenommen hat, kann die Aktenkoffer-Datenbank mit der Master-Konfigurationsdatenbank durch eine Überführungsoperation synchronisiert werden, bei der an reservierten Konfigurationskomponenten durchgeführte Veränderungen in die Master-Konfigurationsdatenbank zurück überführt werden. Im einzelnen überführt eine Überführungsoperation Konfigurationselemente, die von der Master-Konfigurationsdatenbank reserviert wurden, zurück in die Master-Konfigurationsdatenbank, wo diese Veränderungen wiedergegeben werden oder in der Master-Konfigurationsdatenbank gespeichert werden. Eine derartige Operation kann nach Wunsch die Reservierung der Elemente innerhalb der Master-Konfigurationsdatenbank aufheben. Eine Konfigurationsanwendung kann eine Überführungsmitteilung an die Master-Konfigurationsdatenbank zusammen mit den Werten oder den Veränderungen der Elemente, die überführt werden, auf einmal senden, um die Belastung der langsamen Kommunikationsverbindung zu reduzieren. Selbstverständlich können die Operationen des Reservierens und Überführens an einzelnen Elementen von Konfigurationsdaten durchgeführt werden, wie beispielsweise an Bibliothekselementen, Modulen, Knoten, Karten etc., oder können an gesamten Abschnitten eines Baumes durchgeführt werden, wie etwa an einem übergeordneten Element und allen dazugehörigen untergeordneten und weiter untergeordneten Elementen.

Auf Wunsch kann eine Konfigurationsanwendung eine gesamte Aktenkoffer-Datenbank mit der Master-Konfigurationsdatenbank synchronisieren, indem beispielsweise die reservierten Elemente von der Aktenkoffer-Datenbank exportiert werden, die Exportdatei komprimiert wird und die komprimierte Exportdatei an den Knoten, auf dem sich die Master-Datenbank befindet, übertragen wird (das heißt über die langsame bzw. mit niedriger Bandbreite arbeitende Kommunikationsverbindung). An der Master-Datenbank kann eine Anwendung die Exportdatei dekomprimieren, diese Datei in die Master-Datenbank in jeder bekannten oder gewünschten Weise importieren, die aktuelle Struktur der Master-Datenbank in das in der Master-Datenbank verwendete Text- oder Datenformat extrahieren, die Exportdatei komprimieren und die komprimierte Exportdatei zurück zu dem Knoten übertragen, der die Aktenkoffer-Datenbank hat. Die Anwendung in der Aktenkoffer-Datenbank dekomprimiert anschließend die Exportdatei und importiert diese dekomprimierte Datei zurück in die Aktenkoffer-Datenbank. Dieser Prozeß stellt sicher, daß alle Elemente innerhalb der Master-Konfigurationsdatenbank und der Aktenkofferkonfigurationsdatenbank gleich sind.

In **Fig. 3** ist ein Prozeßsteuersystem **100**, das wie vorstehend beschrieben eine Aktenkoffer-Datenbank für Konfigurationsaktivitäten nutzt, dargestellt. Das Prozeßsteuersystem

100 enthält einen Hauptstandort 101 mit Benutzerschnittstellen 102 und 104, die mit einer Steuereinrichtung 106 verbunden sind, welche Steueraktivitäten ausführt. Die Benutzerschnittstellen 102 und 104 sind ferner mit einem Laufzeitdatenserver 108 und einem Konfigurationsdatenbankserver 110 verbunden, die mit einem entfernten Standort 111 über eine langsame Kommunikationsverbindung, wie z. B. eine Satellitenverbindung, eine Modemverbindung, eine zelluläre Verbindung etc. in Kommunikation stehen. Der Hauptstandort 101 enthält ferner eine Master-Konfigurationsdatenbank 112, auf die der Datenbankserver 110 zugreifen kann. In ähnlicher Weise enthält der entfernte Standort 111 Benutzerschnittstellen 114 und 116, die mit einem Laufzeitserver 118 und mit einem Konfigurationsdatenbankserver 120 verbunden sind, die mit dem Hauptstandort 101 über die langsame Verbindung in Kommunikation sind. Der entfernte Standort 111 enthält ferne eine Aktenkoffer-Konfigurationsdatenbank 122. Die Aktenkoffer-Konfigurationsdatenbank 122 kann als eine lokale Datenbank für den entfernten Standort 111 verwendet werden und kann Konfigurationsinformationen über die langsame Verbindung von der Master-Konfigurationsdatenbank 112 an dem Hauptstandort 101 erhalten.

Unter Verwendung einer Konfigurationsanwendung, die Browsing-, Reservierungs- und Synchronisierungsoperationen im Fernbetrieb von einer der Benutzerschnittstellen 114 und 116 bietet, kann ein Techniker Konfigurationsinformationen von der Master-Konfigurationsdatenbank 112 in die Aktenkoffer-Datenbank 122 herunterladen. Als Teil dieses Vorganges kann dem Benutzer Gelegenheit gegeben werden, die Master-Datenbank 112 zu kopieren, die Baumstruktur für die Master-Datenbank 112 zu kopieren, oder die lokale bzw. Aktenkoffer-Datenbank ohne Kopieren einer Struktur von der Master-Datenbank zu erstellen. Zu dieser Zeit kann der Benutzer an dem entfernten Standort 111 auch innerhalb der Master-Konfigurationsdatenbank 112 einen Teil oder die gesamte heruntergeladene Information reservieren, um es zu ermöglichen, daß Veränderungen an dieser Information vorgenommen werden und andere daran gehindert werden, zu versuchen, diese Information innerhalb der Master-Konfigurationsdatenbank 112 zu ändern. Vorzugsweise ergibt die Herunterladeoperation alle erforderlichen Konfigurationsinformationen auf einmal über die langsame Kommunikationsverbindung, was effizienter ist, als diese Daten stückweise über die langsame Verbindung zu erhalten.

Anschließend kann der Benutzer den entfernten Standort 111 oder Elemente desselben unter Verwendung einer Konfigurationsanwendung und der Aktenkoffer-Datenbank 122 an dem entfernten Standort konfigurieren. Auf Wunsch könnte der Benutzer anstelle dessen auch Abschnitte des Hauptstandortes 101 von dem entfernten Standort 111 konfigurieren. Dabei kann die Konfigurationsanwendung, die beispielsweise in der Benutzerschnittstelle 116 ausgeführt wird, auf die Aktenkoffer-Datenbank 122 zugreifen und die Abschnitte der Konfigurationshierarchie oder des Baumes, die in der Aktenkoffer-Datenbank 122 gespeichert sind, anzeigen oder anderweitig nutzen. Wenn der Benutzer die Struktur in der Master-Datenbank 112 sehen möchte, kann der Benutzer eine Synchronisierungsoperation ablaufen lassen, die die Master-Datenbank über die langsame Verbindung nach den gesamten darin enthaltenen Informationen, der Baumstruktur oder nach jedem anderen gewünschten Abschnitt der Konfigurationsinformationen innerhalb der Master-Datenbank 112 abfragt und die Aktenkoffer-Datenbank 122 mit diesen Informationen aktualisiert. Wenn die Konfigurationsaktivitäten an reservierten Elementen beendet sind, kann der Benutzer die an dem bzw. den reservierten Elementen durchgeführten Veränderungen über die langsame Verbindung zurück an die Master-Datenbank 112 übertragen, um zu veranlassen, daß die Änderungen in der Master-Konfigurationsdatenbank 112 gespeichert werden.

Wie **Fig.** 4 zeigt, können eine oder mehrere Aktenkoffer-Datenbanken verwendet werden, um Konfigurationsaktivitäten von nicht verbundenen Workstations auszuführen. Insbesondere zeigt **Fig.** 4 ein Prozeßsteuersystem 130, das einen Hauptstandort 131 enthält, zu dem eine Master-Konfigurationsdatenbank 132, ein Konfigurationsdatenbankserver 133 und eine Benutzerschnittstelle 134 gehören. Entfernte Standorte 135 und 136 enthalten in ähnlicher Weise jeweils einen Datenbankserver 138, eine oder mehrere Benutzerschnittstellen oder Workstations 140, die Konfigurationsanwendungen ausführen können, und eine oder mehrere Aktenkoffer-Konfigurationsdatenbanken 142, die verwendet werden können, um Konfigurationsaktivitäten an jedem der Elemente in der Master-Konfigurationsdatenbank 132 auszuführen. Falls erwünscht, kann die Aktenkoffer-Datenbank 142 innerhalb einer der Benutzerschnittstelleneinrichtungen 140 sein. Ähnlich wie bei dem System von **Fig.** 3 können Benutzer an den Workstations 140 eine Konfigurationsanwendung benutzen, die beispielsweise in einer der Schnittstellen 140 ausgeführt wird, um Verbindung mit der Master-Datenbank 132 aufzunehmen, Konfigurationsinformationen von dieser herunterzuladen und innerhalb der Master-Konfigurationsdatenbank 132 alle Elemente zu reservieren, die verändert werden sollen. Ein derartiges Herunterladen kann über eine langsame Verbindung oder über eine nicht kontinuierliche Verbindung, wie beispielsweise eine Wählverbindung, oder über eine tragbare entfernbare physische Verbindung, wie z. B. einem Laptopcomputer durchgeführt werden. In jedem Fall laden die unterschiedlichen Standorte 135 und 136 einen Teil oder die gesamte Konfigurationsinformation von der Master-Konfigurationsdatenbank 132 in die Aktenkoffer-Datenbank 142. Die Benutzer an den unterschiedlichen entfernten Standorten 135 und 136 können jedoch separate Elemente, die zu verändern sind, reservieren.

Nachdem ein Herunterladevorgang durchgeführt wurde, können Konfigurationsanwendungen an den Workstations 140 durchgeführt werden, um die Elemente zu verändern oder zu modifizieren, die innerhalb jeder Aktenkoffer-Datenbank 142 reserviert wurden, und an einem bestimmten Punkt können Veränderungen an diesen reservierten Elementen unter Verwendung einer Übertragungsprozedur zurück in die Master-Konfigurationsdatenbank 132 hochgeladen werden. Auf diese Weise kann die Aktenkoffer-Datenbank 142 an Workstations 140 verwendet werden, die von dem Hauptnetzwerk 131 völlig getrennt sind oder die über eine langsame Kommunikationsverbindung verbunden sind, so daß beispielsweise ein Techniker Elemente in die Workstation 140 ziehen kann, um an einem anderen Ort zu arbeiten, diese Elemente ändern kann und die neueren Versionen dieser Elemente zu einem späteren Zeitpunkt zurück in die Master-Datenbank 132 übertragen kann. Indem man es ermöglicht, eine Anzahl von unterschiedlichen Aktenkoffer-Datenbanken 142 zur selben Zeit zu erstellen, kann die simultane Entwicklungsarbeit verbessert werden. Insbesondere kann jede Aktenkoffer-Datenbank 142 eine Teilmenge der Konfiguration innerhalb der Master-Konfigurationsdatenbank 132 enthalten, so daß die Konfigurationsoperationen von verschiedenen Technikern logisch geteilt, zugeordnet und vollendet werden können.

Falls gewünscht können neue Konfigurationsdaten periodisch zurück zu der Master-Datenbank 132 übertragen werden, um einen Datenfluß zwischen den unterschiedlichen Aktenkoffer-Datenbanken 142 zu ermöglichen. Bei-

spielsweise kann in der Praxis ein Techniker an einem Ausrüstungsteil arbeiten und ein anderer kann an dem Steuermodul arbeiten, das dieses Ausrüstungsteil betreibt. Der letztere benötigt die endgültige Version der Gerätekonfigurationskomponente, bevor die Steuermodulkonfigurationskomponente vollendet wird. Die endgültige Gerätedefinition kann jedoch aus der Master-Datenbank **132** gezogen werden, nachdem der erste Techniker die Arbeit an der Gerätedefinition beendet hat und den Inhalt dieser Definition zurück in die Master-Datenbank **132** übertragen hat.

Fig. 5 zeigt ein weiteres Beispiel eines Prozeßsteuersystems **145**, in dem Aktenkoffer-Datenbanken vorteilhaft verwendet werden können. Das Prozeßsteuersystem **145** enthält eine Steuerzentrale, die eine Master-Konfigurationsdatenbank **147** hat, die mit einem Datenbankserver **148**, einer Benutzerworkstation **149** und einem Laufzeitserver **150** verbunden ist. Das System **145** enthält ferner eine Betriebsanlagenebene, die einen Datenbankserver **152** und einen Laufzeitserver **154** hat, die mit der Steuerzentrale über eine LAN-Verbindung verbunden sind, wobei es sich um jede Art von LAN-Verbindung einschließlich eines gemeinsam genutzten LAN handeln kann. Die Betriebsanlagenebene enthält ferner eine Aktenkoffer-Datenbank **156**, die beispielsweise mit einem Laptop **158** oder einem anderen Computer, einer Benutzerschnittstelle oder einer Workstation verbunden oder innerhalb derselben angeordnet sein kann, welche beispielsweise von einem Anlagentechniker oder einem Ingenieur verwendet wird, um Veränderungen an der Betriebsanlagenebene durchzuführen. Der Benutzer auf der Betriebsanlagenebene kann eine Konfigurationsanwendung von dem Computer **158** aus ablaufen lassen, um Elemente der Konfigurationsdaten, die in der Master-Datenbank **147** gespeichert sind, über das LAN zu erhalten und wenigstens einige dieser Elemente zu reservieren. Anschließend kann der Benutzer auf der Betriebsanlagenebene, sobald die reservierten Elemente sowie beliebige weitere Elemente, die von einer Konfigurationsanwendung benötigt werden, um Änderungen an der Konfiguration in der Betriebsanlagenebene durchzuführen, in die Aktenkoffer-Datenbank **156** heruntergeladen sind, die Konfigurationsanwendung unter Verwendung der lokalen bzw. Aktenkoffer-Datenbank **156** ablaufen lassen, um Veränderungen an den reservierten Elementen durchzuführen, und anschließend kann er diese Veränderungen über das LAN zurück zu der Master-Datenbank **147** übertragen. Der Prozeß eliminiert die Erfordernisse, zahlreiche Anrufe über das gemeinsam genutzte LAN an die Master-Datenbank **147** durchzuführen, was nicht effizient ist.

Gleichzeitig kann ein Benutzer zuhause einen Computer **160** und eine Aktenkoffer-Datenbank **162** haben, die über einen Datenbankserver **164**, bei dem es sich beispielsweise um ein Modem für Wählverbindungen handeln kann, verbunden sein. Dieser Benutzer kann von der Master-Datenbank **147** über das Modem **164** Konfigurationsdaten erhalten und diese Informationen in der Aktenkoffer-Datenbank **162** speichern, wobei die Elemente reserviert werden, die dieser Benutzer verändern möchte. Der Benutzer zuhause kann anschließend in freier Zeiteinteilung eine Konfigurationsanwendung ausführen, die die Konfigurationsinformationen innerhalb der Aktenkoffer-Datenbank **162** bearbeitet, und Veränderungen nach deren Vollendung über die Modem-Wählverbindung zurück zu der Master-Datenbank **147** übertragen.

Auf Wunsch kann ein Benutzer Informationen, die verschiedene Prozeßsteuersysteme, Standorte etc. betreffen, in derselben Aktenkoffer-Datenbank herunterladen oder plazieren, wenn beispielsweise der Benutzer Veränderungen an zwei verschiedenen Prozeßsteuersystemen durchzuführen wünscht. In dieser Situation können mehrere Aktenkoffer-Datenbanken in einem Computer gespeichert werden und dieser eine Computer kann verwendet werden, um verschiedene Standorte desselben Prozesses zu konfigurieren oder verschiedene Standorte, die zu verschiedenen Prozessen gehören, zu konfigurieren. Typischerweise wird es sich bei einem derartigen Computer um einen Laptop oder einen anderen tragbaren Computer handeln, in dem Konfigurationsanwendungen gespeichert und ausgeführt werden und der eine Konfigurationsdatenbank und zugehörige Datenbankladeund Zugriffsanwendungen enthält.

Falls erwünscht, kann die Master-Datenbank in jedem der Systeme von **Fig.** 1 und 3 bis 5 automatisch ein Revisionsarchiv von Konfigurationselementen speichern, um eine Kontrolle der Quellen für Veränderungen an Konfigurationselementen zu schaffen. In ähnlicher Weise können Elemente in der Aktenkoffer-Datenbank, die modifiziert wurden, visuell gekennzeichnet dargestellt werden, um anzuzeigen, daß die Aktenkoffer-Version modifiziert wurde und daß eine Übertragungsaktivität ausgeführt werden muß.

Anstelle des Vorsehens einer Aktenkoffer-Datenbank kann eine Spiegeldatenbank an entfernten Standorten eines Prozeßsteuersystems unterhalten werden. Die Spiegeldatenbank erweitert das Aktenkoffermodell dadurch, daß zwei Datenbanken, das heißt die entfernte Datenbank und die Master-Datenbank, synchronisiert gehalten werden, was beispielsweise bei einer Festland-/Hochsee-Prozeßsteuerkonfiguration wünschenswert sein kann, wo das Editieren einer Datenbank über eine langsame Verbindung nicht durchführbar ist. Die Spiegeldatenbank kann auch verwendet werden, um die Redundanz zu unterstützen. In der Spiegeldatenbank werden alle Konfigurationselemente, einschließlich beispielsweise Bibliotheks-, Setup- und Steuerdaten, von der Master-Datenbank auf permanenter Basis geschrieben. Ein derartiges Abonnement kann in der in Verbindung mit **Fig.** 7 bis 15 hier beschriebenen Weise umgesetzt werden, falls erwünscht. Ferner wird ein Konfigurationselement automatisch von der Master-Datenbank reserviert, wenn dieses Konfigurationselement in der Spiegeldatenbank editiert wird. Somit sendet ein Editiervorgang in der Spiegeldatenbank automatisch eine Mitteilung über die langsame Kommunikationsverbindung zu der Master-Konfigurationsdatenbank, um dieses Element zu reservieren. Wenn die Reservierungsoperation aufgrund einer Verriegelung dieses Elements innerhalb der Master-Konfigurationsdatenbank nicht durchführbar ist, wird das Editieren in der Spiegeldatenbank verhindert. Zu einem späteren Zeitpunkt wird, nachdem eine Veränderung in der Spiegeldatenbank vorgenommen wurde, das Konfigurationselement von der Spiegeldatenbank zurück in die Master-Konfigurationsdatenbank übertragen. Diese Übertragungsaktivität kann automatisch ausgeführt werden, wenn eine Veränderung durchgeführt wurde, periodisch auf zeitabhängiger Basis, wie etwa alle 10 Minuten, oder von Hand ansprechend auf einen Benutzerbefehl zur Durchführung einer Übertragungsaktivität.

Umgekehrt werden bedingt durch das Abonnementverhältnis der Spiegeldatenbank an der Master-Datenbank Veränderungen an der Master-Konfigurationsdatenbank unmittelbar über die langsame Kommunikationsverbindung zu der Spiegeldatenbank verschoben oder gesendet. Bei einer Spiegeldatenbank können beide Datenbanken unabhängig arbeiten, wenn die Kommunikationsverbindung unterbrochen ist. Modifikationen an den Datenbanken müssen jedoch von Hand abgestimmt werden, wenn die Kommunikationsverbindung wieder aufgenommen wurde. Im allgemeinen können mehrere Spiegeldatenbanken als Abonnenten derselben Master-Datenbank existieren. Aufgrund der

Abonnement-Natur der Verbindung zwischen der Spiegeldatenbank und der Master-Datenbank müssen, sobald die Spiegeldatenbank geschaffen wurde, nur Veränderungen an Daten innerhalb der Spiegeldatenbank und der Master-Datenbank über die langsame bzw. mit geringer Bandbreite arbeitende Kommunikationsverbindung gesendet werden, was effizienter ist, als zu versuchen, Konfigurationsdaten stückweise nach Bedarf von jedem der Benutzer der Spiegeldatenbanken herunterzuladen.

Anstatt eine einzelne Master-Konfigurationsdatenbank und eine oder mehrere Aktenkoffer- oder Spiegeldatenbanken zu haben, die periodisch mit der Master-Konfigurationsdatenbank synchronisiert werden müssen, oder zusätzlich zu dieser Anordnung kann eine Konfigurationsdatenbank über verschiedene Abschnitte eines Prozeßsteuersystems verteilt sein, so daß verschiedene Komponenten der Konfigurationsdaten in verschiedenen physischen Datenbanken gespeichert sind, die an verschiedenen physischen Standorten des Prozeßsteuersystems angeordnet sind. Die verteilten Konfigurationsdatenbanken können verwendet werden, um es zu ermöglichen, daß Konfigurationsinformationen, die zwei oder mehr geographisch getrennte Standorte oder Plätze betreffen, miteinander integriert werden und eine nahtlose Ansicht des gesamten Prozeßsteuersystems gebildet wird, wobei diese nahtlose Ansicht beliebige bzw. alle Prozeßsteuerkonfigurationskomponenten in jedem der unterschiedlichen Standorte oder -plätze enthalten kann.

Fig. 6 stellt ein Prozeßsteuersystem **170** dar, das eine Hierarchie von Konfigurationsdatenbanken hat, die verschiedene logische und/oder physische Orte betreffen. Das Prozeßsteuersystem **170** enthält drei Zonen mit den Namen Zone_A 172, Zone_B 174 und Zone_C 176, zwei Standorte namens Standort_1 180 und Standort_2 182 und ein Unternehmensnetzwerk **184**. Die Zonen **172** und **174** enthalten Konfigurationsdatenbanken **172**a bzw. **174**a und diese Zonen sind beispielsweise über einen Satelliten, ein Modem oder eine andere langsame, mit niedriger Bandbreite arbeitende oder nicht kontinuierliche Kommunikationsverbindung mit dem Standort **180** in Kommunikationsverbindung (wie durch die Linien zwischen den Zonen **172** und **174** und dem Standort **180** dargestellt). Entsprechend enthält die Zone **176** eine Konfigurationsdatenbank **176**a und steht über eine beliebige Kommunikationsverbindung mit dem Standort **182** in Kommunikation. Ferner enthalten die Standorte **180** und **182** Konfigurationsdatenbanken **180**a bzw. **182**a und sind über eine beliebige Kommunikationsverbindung mit dem Unternehmenssystem **184** verbunden. Das Unternehmenssystem **184** enthält auch eine Datenbank **184**a.

Eine Zone, beispielsweise jede der Zonen **172**, **174** und **176**, ist typischerweise eine logische und kann in vielen Fällen eine physische Unterteilung eines großen Steuersystems sein. Somit kann eine Zone allgemein ohne Verbindung zur Außenwelt funktionieren. Eine Zone kann beispielsweise jedes traditionelle Prozeßsteuersystem sein, das miteinander verbundene Einrichtungen an einem bestimmten geographischen Ort hat. Ein Standort, wie etwa jeder der Standorte **180** und **182**, ist eine logische Definition eines Bereiches, einer Region etc., und kann eine beliebige Anzahl von Zonen haben, die zu diesem gehören. Das Unternehmenssystem **184** ist das System der höchsten Ebene in der Konfigurationsdatenbankhierarchie innerhalb des Prozeßsteuersystems **170** und steht mit jedem der Standorte und dadurch mit jeder der Zonen innerhalb des Prozeßsteuersystems **170** in Kommunikationsverbindung. Selbstverständlich könnten andere Standorte, Zonen und Bereiche oder andere logische Einheiten mit der Hierarchie des Prozeßsteuersystems **170** über langsame oder andere Kommunikationsverbindungen verbunden werden und die unterschiedlichen geographischen Orte eines Prozeßsteuersystems können auf andere Weise miteinander verbunden werden, solange die Konfigurationsdatenbank jeder Zone, jedes Standortes, jedes Bereiches etc. von jeder anderen Zone, jedem anderen Standort, Bereich etc. über eine oder eine Reihe von zwei oder mehr Kommunikationsverbindungen zugreifbar ist. Während ferner die verteilte Kommunikationsdatenbankhierarchie von **Fig.** 6 drei Ebenen (Zonen, Standorte und das Unternehmen) enthält, könnten anstelle dessen zwei Ebenen oder vier oder mehr Ebenen verwendet werden.

Wie **Fig.** 6 zeigt, ist jeder Abschnitt des Prozeßsteuersystems **170** mit einer Konfigurationsdatenbank ausgerüstet und die Konfigurationsinformation für das gesamte Prozeßsteuersystem **170** oder die verschiedenen Elemente desselben ist über die Datenbanken **172**a, **174**a, **176**a, **180**a, **182**a und **184**a verteilt. In einer bevorzugten Ausführungsform sind die Konfigurationsdaten oder Konfigurationskomponenten in der Konfigurationsdatenbank für die niedrigste Ebene in der Hierarchie, die aus dem Unternehmen, den Standorten und den Zonen gebildet ist, für die diese Daten alleine benannt sind. So werden beispielsweise Bibliotheksdaten, welche Vorlagen von Einrichtungen und Softwareelementen speichern können, die überall in dem Prozeßsteuersystem **170** verwendet werden können, typischerweise in der höchsten Ebene in der Hierarchie gespeichert und sind in dieser zugänglich, das heißt in der Konfigurationsdatenbank **184**a des Unternehmenssystems **184**. In ähnlicher Weise können Informationen, die die Bibliothekskomponenten und die Systemkonfiguration eines Standortes betreffen, in der Konfigurationsdatenbank des Standortes gespeichert werden, während Konfigurationsinformationen, die zu bestimmten Einrichtungen, Steuermodulen etc. in eine Zoner gehören, in der Konfigurationsdatenbank dieser Zone gespeichert werden. Ferner wird die Steuerkonfigurationsinformation typischerweise in einer Zonenkonfigurationsdatenbank gespeichert, da diese Konfigurationsinformationen physische Einrichtungen in dieser Zone direkt betreffen. Allgemein ausgedrückt ist es das Ziel, jedes bestimmte Teil von Konfigurationsdaten in der Konfigurationsdatenbank der Hierarchie von Datenbanken zu speichern, wo auf diese Konfigurationsdaten am wahrscheinlichsten zugegriffen wird, oder in einer Weise, daß jedes bestimmte Teil von Konfigurationsdaten ohne weiteres von Benutzern an anderen Orten des Prozeßsteuersystems **170** gefunden werden kann.

Bei der in **Fig.** 6 dargestellten verteilten Konfigurationsdatenbankstrategie greift eine Konfigurationsanwendung, die in einer der Zonen, an einem der Standorte, in einem der Bereiche etc. des Prozeßsteuersystems **170** abläuft, auf Konfigurationsdaten von einer oder mehreren Konfigurationsdatenbanken zu, die an demselben Standort, in derselben Zone etc. und/oder an anderen Standorten, in anderen Zonen etc. angeordnet sein können, und greift auf diese Daten nach Bedarf zu, um den gegenwärtigen Status des Prozeßsteuersystems **170** darzustellen oder andere Konfigurationsoperationen auszuführen. Es versteht sich, daß eine Konfigurationsanwendung, die von einer der Zonen, einem der Standorte oder dem Unternehmenssystem von **Fig.** 6 ausgeführt wird, in der Lage ist, auf die Konfigurationsdaten zuzugreifen, die in jeder der verschiedenen Konfigurationsdatenbanken gespeichert sind, um eine Darstellung der aktuellen Konfiguration jedes Teiles oder des gesamten Prozeßsteuersystems **170** zu geben, und in der Lage sein kann, jedes der Elemente innerhalb jedes Standortes, innerhalb jeder Zone etc. des Prozeßsteuersystems **170** neu zu konfigurieren. Eine verteilte Konfigurationsdatenbank, wie etwa die in **Fig.** 6 gezeigte, hat die Tendenz, die Menge der Kommunikation zu vermindern, die über die langsame Kommunikationsverbin-

dung durchgeführt werden muß, da typischerweise Benutzer an einem Standort oder in einer Zone wahrscheinlicher die Konfigurationsdaten dieses Standortes oder dieser Zone betrachten, benutzen oder verändern, als diejenigen eines anderen Standortes oder einer anderen Zone, was bedeutet, daß die meisten Lesevorgänge aus und Schreibvorgänge in die Konfigurationsdatenbank innerhalb eines Standortes oder einer Zone von Benutzerschnittstellen innerhalb dieses Standortes oder dieser Zone auftreten, während weniger derartige Lesevorgänge und Schreibvorgänge von anderen Zonen oder Standorten über eine oder mehrere langsame Kommunikationsverbindungen kommen. Es ist jedoch die gesamte Konfigurationsinformation für jeden Teil des Prozeßsteuersystems **170** für Benutzerschnittstellen oder andere Einrichtungen in jedem anderen Teil des Prozeßsteuersystems **170** verfügbar.

Um die Kommunikation zwischen den verschiedenen Zonen, Standorten etc. zu ermöglichen, kann jede Datenbank innerhalb des Systems Informationen speichern, die die allgemeine Einstellung der Konfigurationshierarchie betreffen. Beispielsweise kann jede Datenbank einige Objektwurzeln speichern, die für die Konfiguration in allen Orten gemeinsam sind, und Zeiger speichern, welche die bestimmte Datenbank angeben, die weitere Konfigurationsinformationen über diese Wurzeln hat, oder die die nächste Datenbank angeben, die für diese Information abzufragen ist. So kann ein Benutzer in Zone_A 172 versuchen, auf ein Wurzelobjekt zuzugreifen und untergeordnete Elemente dieser Wurzel anfordern. Ansprechend darauf kann die Konfigurationsdatenbank **172**a die Datenbank finden, welche die untergeordneten Elemente speichert, indem ein Zeiger für die ausgewählte Wurzel betrachtet wird, und dieser Zeiger kann angeben, daß auf die Daten für diese Wurzel von der Unternehmensdatenbank **184**a zugegriffen werden muß oder daß auf die Datenbank **180**a von Standort_1 zugegriffen werden sollte, um weitere Zeiger für diese Daten zu erhalten. Die Datenbank **172**a kann dann einen Auslesevorgang der Datenbank **180**a über eine langsame Verbindung veranlassen, was veranlassen kann, daß die Datenbank **180**a den Lesevorgang der Unternehmensdatenbank **184**a durchführt. Selbstverständlich ist es möglich, daß die Unternehmensdatenbank **184**a auf die Datenbank **182**a von Standort_2 zugreift, welche möglicherweise auf die Datenbank **176**a von Zone_C für diese Daten zugreifen muß. Alternativ können Browser- oder Suchanwendungen an jeder Datenbank vorgesehen sein, um beispielsweise die Datenbank **172**a von Zone_A, die Datenbank von Standort_1 oder andere Datenbanken zu durchsuchen, um den Ort eines bestimmten Teiles von Konfigurationsdaten innerhalb des verteilten Konfigurationssystems **170** zu finden.

Auf Wunsch kann jede Datenbank innerhalb des Systems **170** eine lokale Kopie von Daten speichern, die ursprünglich an einem anderen Standort, in einer anderen Zone etc. gespeichert sind. Insbesondere wenn eine Anfrage nach Daten, die nicht in einer bestimmten Datenbank gespeichert sind, erhalten wird, kann die Datenbank diese Daten von einer anderen Datenbank abrufen und bei Abrufen dieser Daten diese lokal zur Benutzung durch die Benutzer speichern, die direkt mit dieser Datenbank (wie etwa die Benutzer an demselben Standort und in derselben Zone) verbunden sind, oder sie ansprechend auf Anforderungen von anderen Datenbanken innerhalb der Datenbankhierarchie weitersenden. Somit kann in vorstehend beschriebenem Beispiel jede der Datenbanken **180**a, **184**a und **182**a eine lokale Kopie von Konfigurationsdaten speichern, die ursprünglich in der Datenbank **176**a von Zone_C gespeichert sind, da diese Daten von der Datenbank **172**a der Zone_A oder einem Benutzer in Zone_A angefordert wurden.

Da auf einige oder alle Konfigurationsdaten für eine bestimmte Konfigurationsansicht, die auf einer beliebigen Benutzerschnittstelle angezeigt wird, über eine oder mehrere langsame Kommunikationsverbindungen zugegriffen wird, ist es möglich, in eine Situation zu geraten, in der Konfigurationsdaten, die auf einer bestimmten Benutzerschnittstelle angezeigt werden, basierend auf Veränderungen, die an der Konfigurationsinformation durch einen anderen Benutzer durchgeführt wurden, sei es an demselben Standort oder in derselben Zone oder anderswo, überaltert bzw. nicht mehr aktuell sind. Während jede Konfigurationsanwendung periodisch auf die entsprechende Konfigurationsdatenbank zugreifen könnte, um die Konfigurationsinformation, die auf einem Bildschirm dargestellt wird, zu aktualisieren, würde diese periodische Aktualisierung bzw. Auffrischung das Ausmaß der Kommunikation, die über die langsame Kommunikationsverbindung durchgeführt werden muß, beträchtlich erhöhen und würde ferner die Zahl der Lesevorgänge aus den Konfigurationsdatenbanken erhöhen, was die Kommunikation innerhalb des Prozeßsteuersystems **170** verlangsamt und die Belastung jeder der Konfigurationsdatenbanken darin erhöht.

Zur Lösung des Problems kann jede Konfigurationsanwendung, die auf Konfigurationsdaten zugreift, die geeignete Konfigurationsdatenbank für jede Konfigurationskomponente, die auf dem Bildschirm dargestellt (oder anderweitig verwendet) wird, abonnieren und die Konfigurationsdatenbank, die diese Daten speichert, benachrichtigt automatisch jede Abonnentenkonfigurationsanwendung (Client) über Veränderungen, die an der Information innerhalb der Konfigurationsdatenbank vorgenommen wurden. Diese Veränderungen können anschließend automatisch an den Client gesendet werden und beispielsweise auf einem Benutzerbildschirm dargestellt werden, oder die Veränderungen können von dem Benutzer angefordert werden. Auf diese Weise ist die Konfigurationsinformation, die von jeder Konfigurationsanwendung verwendet wird, beispielsweise jedes Teil der Informationen, die auf jedem Bildschirm dargestellt werden, ungeachtet dessen, wo die Client-Anwendung, welche diese Information verwendet, innerhalb des Prozeßsteuersystems **170** angeordnet ist, aktuell.

Wenn eine Client-Anwendung bestimmte Stücke einer Konfigurationsinformation nicht mehr benötigt, beispielsweise wenn ein Benutzer nicht länger ein bestimmtes Stück der Konfigurationsinformationen in einer Konfigurationshierarchie betrachten möchte, löst die Konfigurationsanwendung selbstverständlich das Abonnement mit der Konfigurationsdatenbank, in der dieser Teil der Information gespeichert ist, so daß die Benachrichtigungen über Veränderungen oder Aktualisierungen nicht weiter zu der abonnierenden Konfigurationsanwendung gesendet werden. Auf diese Weise findet die Konfigurationsanwendung die Konfigurationsdatenbank, in der jeder Teil der Konfigurationsinformation innerhalb beispielsweise des Baumes **65** von **Fig. 2** gespeichert ist, ruft diese Information erforderlichenfalls durch eine oder mehrere langsame Kommunikationsverbindungen ab, abonniert jeden Teil der Information darin, um so Aktualisierungen, die an dieser Information innerhalb der Konfigurationsdatenbank, in der diese Information gespeichert ist, zu empfangen, und zeigt nach Erhalt von Aktualisierungen der dargestellten Information (welche durch einen anderen Benutzer verursacht sein können, der Veränderungen an der Konfiguration des betrachteten Systems vornimmt) die neue bzw. aktualisierte Information an. Wenn ein Benutzer oder eine Bedienungsperson einen Abschnitt des Konfigurationsbaumes, der dargestellt wird, tiefer hinab verfolgt, greift die Konfigurationsanwendung auf die neue Information zu, abonniert diese Information und zeigt diese

Information in derselben Weise an. Wenn ein Teil der angezeigten Information von dem Benutzerbildschirm entfernt wird, löst die Konfigurationsanwendung das Abonnement der Daten, die fallengelassen wurden.

Auf Wunsch können alle Konfigurationsinformationen, die erhalten wurden (beispielsweise am Bildschirm einer bestimmten Benutzerschnittstelle betrachtet wurden), in einem lokalen Cache oder Speicher gespeichert werden, der mit einer Konfigurationsanwendung verbunden ist, die abläuft, oder können in der Konfigurationsdatenbank innerhalb der Zone, des Standorts etc. gespeichert werden, in der die Anwendung abläuft, so daß dann, wenn eine Verbindung zu der entsprechenden Datenbank unterbrochen wird, der oder die Benutzer an dem Standort, in der Zone etc. diese Konfigurationsinformationen weiterhin aus dem lokalen Cache entnehmen können. Diese Informationen können jedoch in einer Weise dargestellt werden, die angibt, daß diese Informationen aus dem lokalen Cache stammen (beispielsweise grau hinterlegt dargestellt), so daß der Benutzer weiß, daß diese Informationen möglicherweise nicht aktuell sind. Die Verwendungen eines derartigen lokalen Cache zum Speichern von Konfigurationsdaten, die bereits über eine oder mehrere langsame Verbindungen erhalten wurden, ermöglicht es einem Benutzer, jede Konfigurationsinformation zu betrachten, die bereits betrachtet bzw. auf die bereits von einem Benutzer zugegriffen wurde, wenn eine Kommunikationsverbindung oder -leitung unterbrochen wird. Auch wenn der Benutzer Informationen zu betrachten wünscht, die von seinem Bildschirm bereits wieder fallengelassen wurden, können diese Informationen aus dem lokalen Cache entnommen werden, womit sie rascher erscheinen, während die Verbindung zu der geeigneten Konfigurationsdatenbank für diese Informationen wiederhergestellt werden kann, um diese Informationen zu aktualisieren.

Während **Fig.** 6 jeden dieser Standorte und jede dieser Zonen als durch eine langsame Verbindung verbunden darstellt, ist ein derartiger Aufbau nicht unbedingt notwendig. Beispielsweise kann die Konfigurationsdatenbank für einen Standort in einer Zone angeordnet sein und direkt von dieser Zone zugreifbar sein, während sie über eine langsame Kommunikationsverbindung für andere Zonen zugreifbar ist, die zu diesem Standort gehören. Um beispielsweise eine verteilte Konfigurationsdatenbank wie die in **Fig.** 6 gezeigte zu schaffen, kann ein Benutzer in Zone_A 172 eine Standortdatenbank (das heißt die Datenbank **180**a für den Standort_1) auf demselben Gerät wie die Datenbank **172**a für die Zone_A oder auf einem anderen Gerät, das mit der Datenbank **172**a für die Zone_A über eine direkte Hochgeschwindigkeitsverbindung verbunden ist, einrichten. Alternativ kann die Datenbank **180**a für den Standort_1 mit der Datenbank **172**a für die Zone_A über eine langsame Verbindung verbunden sein. Der Benutzer veröffentlicht einige oder alle Bibliotheken und Einrichtungsdaten in Zone_A für die Datenbank **180**a von Standort_1 und/oder für die Datenbank **184**a des Unternehmenssystems. Anschließend kann die Datenbank **172**a in Zone_A 172 automatisch eine oder mehrere der Konfigurationselemente innerhalb der Datenbank **180**a des Standorts_1, der Unternehmensdatenbank **184**a etc. abonnieren, wenn ein Benutzer oder eine Konfigurationsanwendung innerhalb der Zone_A 172 Zugriff auf diese Elemente benötigt oder wünscht. Nach Wunsch kann jede Konfigurationsdatenbank des Systems **170** immer einige oder alle der Konfigurationsinformationen innerhalb einer oder mehrerer der anderen Konfigurationsdatenbanken des Systems abonnieren.

Desweiteren kann ein Benutzer in Zone_B 174 die Datenbank **180**a von Standort_1, die von dem Benutzer in Zone_A geschaffen wurde, als die Standortdatenbank für die Zone_B 174 auswählen. Der Benutzer in Zone_B 174 abonniert dann einige oder alle Bibliotheks- und Setup-Daten in der Datenbank **180**a für den Standort_1, was zu Konflikten führen kann, sofern ein Element mit demselben Namen in der Datenbank **180**a für den Standort_1 und der Datenbank **174**a für die Zone_B existiert. Eine Konfigurationsanwendung kann den Benutzer in Zone_B 174 über dieses Problem unterrichten und der Benutzer in Zone_B 174 kann die Option angeboten bekommen, ein lokales Element neu zu benennen, bevor Elemente in der Datenbank **180**a von Standort_1 abonniert werden, oder das lokale Element zu überschreiben. Zusätzlich kann der Benutzer in Zone_B 174 die Elemente für die Datenbank **180**a von Standort_1 veröffentlichen und anschließend wird Zone_A 172 über die Hinzufügungen zu der Datenbank **180**a von Standort_1, die Zone_A abonniert hat, benachrichtigt, und kann anschließend diese zusätzliche Konfigurationselemente abonnieren. Selbstverständlich wird die Unternehmens-/Standorthierarchie in ähnlicher Weise konfiguriert, um dadurch die Kommunikation zwischen den Standorten **180** und **182** und zwischen den Zonen **172**, **174** und **176** zu ermöglichen.

Um eine Verbindung zwischen Zone_A 172 und Zone_C 176 zu schaffen, wenn beispielsweise ein Benutzer in Zone_A 172 Konfigurationskomponenten zu betrachten wünscht, die in der Datenbank **176**a von Zone_C 176 gespeichert sind, veröffentlicht die Datenbank **176**a von Zone_C diese Daten für die Datenbank **182** von Standort_2, welche diese Informationen abonniert und welche diese Informationen für die Unternehmenssystemdatenbank **184**a veröffentlicht, welche diese Informationen von Standort_2 182 abonniert hat. Standort_1 180 abonniert diese Informationen von der Unternehmensdatenbank **184**a, während Zone_A 172 diese Informationen von der Datenbank **180** von Standort_1 abonniert. Die eingerichtete Abonnentenbeziehung kann anschließend verwendet werden, um einen Client oder Benutzer in Zone_A 172 über Veränderungen der Konfigurationskomponenten, die in der Datenbank **176**a von Zone_C gespeichert sind, zu benachrichtigen. Wie vorstehend angegeben kann eine Zone, ein Standort etc. Elemente individuell oder einen gesamten Verzweigungsbaum von Objekten einschließlich übergeordneten Elementen und allen zugehörigen untergeordneten Elementen abonnieren. Das Abonnement kann auch das Übernehmen einer Kopie der Elemente, das Hinzufügen dieser Elemente zu der Konfigurationsdatenbank der lokalen Zone und das Schaffen einer fortdauernden Kommunikationsbeziehung zwischen den veröffentlichten Elementen und den lokalen Kopien umfassen. Die veröffentlichende Datenbank kann die Abonnementen verfolgen, während die abonnierenden Datenbanken verfolgen, wo die ursprüngliche Information gespeichert ist. Diese Quellen-/Bestimmungsortinformation kann verwendet werden, um Reservierungen von Elementen innerhalb einer Konfigurationsdatenbank zu leiten und Veränderungen von entfernten Orten aus vorzunehmen. Entsprechend kann diese Information verwendet werden, um die Veröffentlichungs-/Abonnentenbeziehung neu herzustellen, wenn eine Datenbank aus der Sicherungskopie wieder aufgebaut werden muß.

Da die Kommunikation zwischen Zonen nicht immer aufgrund des Vorhandenseins der langsamen Kommunikationsverbindung zwischen den Zonen garantiert werden kann, funktionieren die Zonen natürlich weiterhin unabhängig, wenn die Kommunikationsverbindungen unterbrochen sind. Wenn die Kommunikation wieder hergestellt ist, müssen jedoch gegebenenfalls Konflikte von Hand gelöst werden.

Es versteht sich, daß die Kommunikation zwischen Zonen, Standorten und dem Unternehmen auf der Basis von Veröffentlichung und Abonnement durchgeführt werden

kann, wobei die Konfigurationsdatenbank, welche die Masterkopie des Konfigurationselementes hat, dieses Element veröffentlicht, wenn ein oder mehrere Abonnenten dieses Element abonnieren. Die Abonnenten können beispielsweise Konfigurationsanwendungen sein, die in Benutzerschnittstellen (beispielsweise Workstations oder anderen Computern) in verschiedenen Standorten oder Zonen ausgeführt werden, können Datenbanken innerhalb dieser Standorte oder Zonen sein, welche Datenbanken von Konfigurationsanwendungen verwendet werden, die in diesen Zonen ausgeführt werden, oder können beliebige andere Einrichtungen oder Anwendungen sein, welche die Konfigurationsdaten benutzen. Die Elemente innerhalb einer Datenbank, die von anderen Standorten oder Zonen abonniert werden können, werden als gemeinsam genutzte Objekte bezeichnet. Selbstverständlich können unterschiedliche Arten von Operationen an gemeinsam genutzten Objekten durchgeführt werden, wie etwa 1.) Durchsuchen, wobei eine Abonnentenkonfigurationsanwendung die Liste der gemeinsam genutzten Objekte, die von dem Veröffentlicher verfügbar sind, sichtet, 2.) Abonnieren, wobei eine Abonnentenkonfigurationsanwendung eine lokale Kopie eines gemeinsam genutzten Objektes zur Verwendung bei Konfigurationsaktivitäten am Ort des Abonnenten anfordert, 3.) Kopieren in ein lokales Objekt, wobei eine Abonnentenkonfigurationsanwendung ein Element in ein nicht gemeinsam genutztes lokales Objekt oder eine entsprechende Datenbank kopiert, 4.) Löschen, wobei eine Abonnentenkonfigurationsanwendung das gemeinsam genutzte Objekt in dem Veröffentlicher löscht, 5.) Lösung des Abonnements, wobei eine Abonnentenkonfigurationsanwendung die Abonnementverbindung zwischen der lokalen Kopie des Konfigurationselements bei dem Abonnenten und der Veröffentlicherkopie unterbricht, 6.) Veränderungen abholen, wobei die Abonnentenkonfigurationsanwendung die neueste Version des gemeinsam genutzten Elements von dem Veröffentlicher abruft, 7.) Reservieren, wobei die Abonnentenkonfigurationsanwendung ein gemeinsam genutztes Objekt in der Veröffentlicherdatenbank verriegelt, um zu verhindern, daß dieses Objekt durch einen anderen Benutzer verändert wird, und 8.) Übertragen, wobei die Abonnentenkonfigurationsanwendung Veränderungen, die an einem reservierten Element durchgeführt wurden, zurück in die Veröffentlicherdatenbank schiebt. Diese Operationen können durch Übersenden von Mitteilungen zwischen dem Abonnenten und dem Veröffentlicher durchgeführt werden und die Ausführung von Software an dem Abonnentenstandort und dem Veröffentlicherstandort, um die Mitteilungen zu verarbeiten und geeignete Aktionen wie vorstehend beschrieben auszuführen.

Aus dem Blickwinkel der Konfiguration gibt es verschiedene Aktivitäten zwischen Zonen, die durchgeführt werden können, einschließlich das gemeinsame Nutzen von Setup- und Bibliotheksdaten zwischen Zonen, das Durchsuchen oder Betrachten von Setup-, Bibliotheks- und Konfigurationsdaten aus einer anderen Zone, die Konfigurierung von Verweisen zwischen den Zonen und die Koordination von Namensraum (Namespace) zwischen den Zonen. In einer Ausführungsform werden Konfigurationsobjekte oder -elemente von den unmittelbar übergeordneten Elementen innerhalb der Hierarchie gemeinsam genutzt. Beispielsweise nutzen die Zonen **172** und **174** eine zusammengesetzte Definition von dem Standort **180** gemeinsam und alle Zonen nutzen einen gemeinsamen Aufzählungssatz von dem Unternehmenssystem **184** gemeinsam. Die Zonen **172** und **174** abonnieren die Definitionen in den Standort **180**, während die Zone **176** die Definitionen in dem Standort **182** abonniert. Entsprechend abonnieren die beiden Standorte **180** und **182** gemeinsame Definitionen in dem Unternehmenssystem **184**.

Durchsuchungs-Dienste bzw. Browser sind in einer Konfigurationsanwendung vorgesehen, um einen Server in der Unternehmens-/Standort-/Zonenhierarchie zu finden und die Konfigurationsdatenbank zu durchsuchen, die zu einem dieser Orte gehört. Durchsuchungsdienste werden verwendet, um ein Abonnement an gemeinsam genutzten Objekten einzurichten und die Verweise zwischen den Zonen zu konfigurieren. Insbesondere wird die Durchsuchungsinformation von einem entfernten Server auf Anfrage eines Benutzers abgefragt und lokal im Cache gespeichert. Die Durchsuchungswurzeln können durch Absuchen des Netzwerkes erfaßt werden oder können unter Verwendung jeder anderen gewünschten oder bekannten Durchsuchungstechnik gefunden werden. Das Wissen über erfaßte oder konfigurierte Durchsuchungswurzeln kann dauerhaft sein und bleibt so über einen Einschaltzyklus im Cache gespeichert. Das Vorhandensein dieser Konfigurationsobjektwurzeln ermöglicht es einem Benutzer, andere Informationen, die zu diesen Wurzeln gehören, bei der Durchsuchung zu finden und den Ort oder den Pfad zu dieser Information innerhalb der verteilten Konfigurationshierarchie zu finden.

Verweise zwischen Zonen können konfiguriert werden, indem der Verweis direkt eingegeben wird, oder indem beim Durchsuchen ein Attribut in einer entfernten Datenbank aufgefunden wird. Verweise können die Form von beispielsweise "//ZonenName/MarkiertesElement/. . ./Attributname" oder jede andere gewünschte Form annehmen. Es versteht sich, daß Zonennamen innerhalb einer Unternehmenshierarchie einzigartig sein müssen, um Verweise zwischen Zonen praktikabel zu machen. Innerhalb jeder Zone ist ein Namensraum für markierte Elemente vorhanden und das Konfigurationssystem kann die Einzigartigkeit von Markierungen innerhalb einer Zone zwangsweise durchsetzen. Somit kann jede Konfigurationsanwendung ein Tool haben, welches bestimmt, ob eine Markierung innerhalb eines Standortes oder innerhalb des Unternehmens zu einem gegebenen Moment einzigartig ist, um die Einzigartigkeit bei der Benennung innerhalb der verteilten Konfigurationsdatenbank zwangsweise durchzusetzen.

Auf Wunsch kann die Sicherheit zwischen Zonen in jeder gewünschten Weise geschaffen werden und ein Zonenadministrator kann die Privilegien für den Zugriff auf gemeinsam genutzte Objekte für einen Benutzer, für eine Gruppe etc. gewähren. Diese Privilegien können die Fähigkeit einschließen, gemeinsam genutzte Objekte zu durchsuchen und zu abonnieren, die Fähigkeit, Veränderungen von einem Veröffentlicher an Objekten oder Konfigurationselementen abzuholen, und die Fähigkeit, Veränderungen zu reservieren, zu modifizieren und zu übertragen.

Ferner können auf Wunsch unterschiedliche Zonen verschiedene Sprachen nutzen, was beispielsweise in Europa von Vorteil sein kann, wo Firmen in Regionen mit vielen unterschiedlichen Sprachen tätig sind. In diesem Fall können Aufzählwerte zwischen Zonen als numerische Werte oder als ein lokaler Datenstring (wie z. B. ein Wort) weitergeleitet werden, welche zusammen mit einem numerischen Wert, der über das ganze Prozeßsteuersystem für dasselbe Wort oder denselben Satz in unterschiedlichen Sprachen gemeinsam ist, angezeigt werden können. Auf diese Weise kann ein Benutzer den Aufzählungswert, den Befehl etc. an seiner entsprechenden Zahl erkennen, was ohne weiteres die Umwandlung von einer Sprache in die andere ermöglicht. In jedem Fall erfordert das Durchsuchen einer Datenbank, die eine andere Sprache benutzt, daß der entsprechende Zeichensatz in der Durchsuchungseinrichtung oder Zone installiert ist. In einer Ausführungsform sind die Verweise zwischen den Zonen in der Sprache der entfernten Zone konfi-

guriert und Unicode-Strings werden in dem ganzen System ausschließlich verwendet, um das Erfassen oder Umwandeln jedes Zeichens in jede Sprache, die von Unicode unterstützt wird, zu ermöglichen. In diesem Fall kann eine einzige Datenbank das Speichern von Strings von mehreren Sprachen bewältigen und Datenbankexportdateien werden im Unicodeformat ausgetauscht. Mit anderen Worten werden Dateien von verschiedenen Orten in jede Datenbank unter Verwendung des Unicodeformats importiert.

Um sicherzustellen, daß Modifikationen ohne Störung durchgeführt werden können, können Modifikationen an gemeinsam genutzten Daten vorbehaltlich eines Reservierungs-/Übertragungsvorganges durchgeführt werden, der vorstehend beschrieben wurde. In diesem Fall muß der modifizierende Benutzer, bevor ein Element revidiert werden kann, dieses Element in der Zone des modifizierenden Benutzers reservieren. Nur ein Benutzer oder eine Zone können ein Element zu einem Zeitpunkt jeweils reservieren. Nachdem ein Element reserviert wurde, werden Veränderungen an dem Element in der lokalen Datenbank durchgeführt, die dieses Element reserviert hat, wobei derartige Veränderungen als ein Resultat einer neuen Konfigurationsaktivität durch den Benutzer, der das Element reserviert hat, durchgeführt werden. Anschließend wird das Element bzw. die Gruppe von Elementen, die reserviert wurde und modifiziert wurde, zurück in die Konfigurationsdatenbank übertragen, wo diese Daten als Master-Datensatz gespeichert sind. Ein derartiger Reservierungs-/Übertragungsvorgang stellt sicher, daß nicht zwei Benutzer versuchen können, dasselbe Element zur selben Zeit zu verändern. Der zweite Benutzer, der versucht, das Element zu reservieren, wird anstelle dessen daran gehindert, Modifikationen durchzuführen, bis der erste Benutzer, der das Element reserviert hat, die Veränderungen an dem Element zurück in die Konfigurationsdatenbank übertragen hat, in der die Masterversion dieses Elements gespeichert ist.

Allgemein können zwei Modi des Abonnements verwendet werden. In dem ersten Modus, der hier als sicherer Abonnementmodus bezeichnet wird, werden neue gemeinsam genutzte Elemente automatisch in eine Abonnementzone oder einen Standort gezogen, während Modifikationen an Elementen bzw. Löschungen von Elementen, die bereits vorhanden sind, von Hand gezogen werden, d. h. es ist ein direkter Befehl von dem Benutzer erforderlich, um das Hereinziehen der veränderten oder gelöschten Information durchzuführen. In diesem Fall kann der Benutzer über eine Veränderung oder eine Löschung an einer Konfiguration, die das Benutzersystem abonniert hat, benachrichtigt werden, und kann gefragt werden, ob die Konfigurationsanwendung dieser Veränderung in die lokale Datenbank ziehen soll. Alternativ können alle Modifikationen automatisch gezogen werden, so daß alle Veränderungen an der gemeinsam genutzten Datenbank in die abonnierenden Datenbanken ohne Eingreifen des Benutzers importiert werden. Es sei angemerkt, daß in dem sicheren Abonnementmodus eine Mitteilung einer Veränderung nicht durch die gesamte Hierarchie durchgängig ist. Wenn somit beispielsweise ein Unternehmenselement revidiert wird, werden die abhängigen Standorte benachrichtigt. Erst wenn jedoch die Veränderungen tatsächlich in einen Standort gezogen werden, werden die zu diesem Standort gehörigen Zonen über die Veränderungen benachrichtigt. Als Resultat kann der Veröffentlicher, das heißt die Datenbank, welche die Masterkopie dieses Konfigurationselements speichert, verfolgen, ob ein Abonnent die neueste Revision eines Elements genommen hat, und diese Information kann verwendet werden, um die Veröffentlicher/Abonnentenbeziehung neu zu synchronisieren, wenn eine der Datenbanken aus der Sicherungskopie wieder aufgebaut werden muß.

Ferner versteht sich, daß dann, wenn ein Abonnent Veränderungen nicht abholt, dies zu Unterschieden in der Konfigurationshierarchie in niedrigeren Ebenen in der Konfigurationsdatenbankhierarchie führen kann. Wenn beispielsweise das Unternehmenssystem **184** die Standorte **180** und **182** von **Fig.** 6 über eine Veränderung benachrichtigt und wenn der Standort_2 182 diese Veränderung abholt, während der Standort_1 180 dies nicht tut, so ist die Konfigurationshierarchie, die an den Standorten **180** und **182** zu betrachten ist, verschieden. Wenn ferner der Standort_2 182 die Veränderung abholt, wird eine Veränderungsmitteilung an die Abonnenten in der Zone_C 176 gesendet, welche diese Veränderungen abholen können oder zumindest wissen, daß eine Veränderung aufgetreten ist. Die Abonnenten in der Zone_A 172 und Zone_B 174 wissen jedoch nichts von der Veränderung, da der Standort_1 180 die Veränderung nicht abgeholt hat und daher Zone_A 172 und Zone_B 174 die Veränderungen nicht abholen können. In diesem Fall ist die Konfigurationshierarchie, die in den Zonen **172** und **174** verfügbar ist, anders als die, die in der Zone **176** verfügbar ist. Auf Wunsch kann eine Konfigurationsanwendung vorgesehen werden, welche die Liste von stromaufwärts und stromabwärts verlaufenden Abhängigkeiten anzeigt, um dem Benutzer zu erleichtern zu erkennen, was hinsichtlich Veränderungen geschieht. Wenn der Benutzer versucht, ein Element abzuholen, kann die Konfigurationsanwendung mangelnde Übereinstimmungen hinsichtlich einer stromaufwärts verlaufenden Revision erfassen und den Benutzer fragen, ob diese Elemente auch abgeholt werden sollten.

Unter Bezug auf **Fig.** 7 bis 15 wird ein System für den effizienten Zugriff und die effiziente Verteilung von Daten von einer Datenbank innerhalb des Prozeßsteuersystems beschrieben. Allgemein ausgedrückt nutzt das Datenbankzugriffssystem einen gemeinsam genutzten Cache, um in der Datenbank gespeicherte Daten für einen oder mehrere Abonnenten (auch als Clients bezeichnet) dieser Daten zu veröffentlichen und Mitteilungen über Aktualisierungen oder Veränderungen an die Abonnenten oder Clients abzugeben, wenn Veränderungen an den Daten innerhalb der Datenbank auftreten. Es versteht sich, daß das Datenbankzugriffssystem, das hierin beschrieben wird, in jeder der Datenbanken innerhalb eines Prozeßsteuersystems verwendet werden kann, wie etwa jeder der Konfigurationsdatenbanken **30** und **54** in **Fig.** 1, jeder der Master-Konfigurationsdatenbanken von **Fig.** 3 bis 5 oder jeder der Konfigurationsdatenbanken von **Fig.** 6. Entsprechend kann der Abonnent oder Client der Daten in einer Datenbank, auf die zugegriffen wird, jede andere Konfigurationsdatenbank, jede Konfigurationsanwendung, die in einer beliebigen Benutzerschnittstelle innerhalb des Systems ausgeführt wird, oder jede andere Anwendung sein, welche die Konfigurationsdaten benutzt, und diese Clients können mit der Datenbank, auf die zugegriffen wird, beispielsweise unter Verwendung einer direkten bzw. Hochgeschwindigkeitskommunikationsverbindung, einer langsamen bzw. mit niedriger Bandbreite arbeitenden Kommunikationsverbindung oder einer nicht kontinuierlichen Kommunikationsverbindung verbunden sein. So können beispielsweise die in **Fig.** 7 bis 5 beschriebenen Clients in jeder der Zonen oder an jedem der Standorte von **Fig.** 1 und 3 bis 6 angeordnet sein.

Allgemein ausgedrückt kommuniziert jeder Client mit einem Datenbankserver, der mit einer Konfigurationsdatenbank verbunden ist, die die Konfigurationskomponenten speichert, welche der Client abonniert. Am Beginn des Abonnierungsprozesses wird ein Clientthread innerhalb des Datenbankservers für jeden verschiedenen Client geschaffen, der Verbindung zu der Konfigurationsdatenbank hat,

und dieser Thread bietet Zugang zu einer oder mehreren bestimmten Komponenten innerhalb der Konfigurationsdatenbank. Anschließend werden Veränderungen an den Komponenten innerhalb der Konfigurationsdatenbank erkannt und Mitteilungen über diese Veränderungen und nach Wunsch die Veränderungen selbst werden zu jedem Client zurückgesendet, der diese Konfigurationskomponenten abonniert hat. Wenn ein Client das Abonnement dieser Konfigurationskomponente löst, werden keine Mitteilungen über Veränderungen für diese Komponente an den Client gesendet. Ein Client kann auch einen Clientthread verwenden, um die Komponente zu verriegeln, zu reservieren, zu übertragen oder anderweitig Veränderungen daran vorzunehmen.

Es versteht sich, daß die Verwendung des Begriffes "Thread" sich auf einen Verarbeitungspfad oder einen Verarbeitungsvorgang bezieht, der von einem Prozessor innerhalb des Datenbankservers ausgeführt wird, und eine bekannte Technik zur Durchführung von Parallelverarbeitung darstellt. Allgemein ausgedrückt, kann ein Prozessor mehrere Threads gleichzeitig durch versetztes Bearbeiten von Tasks ausführen, die zu den verschiedenen Threads gehören, die durchgeführt werden. Beispielsweise kann der Prozessor in dieser Reihenfolge den ersten Schritt, der zu einem ersten Thread gehört, den ersten Schritt, der zu einem zweiten Thread gehört, den ersten Schritt, der zu einem dritten Thread gehört, anschließend den zweiten Schritt, der zu dem ersten Thread gehört, den zweiten Schritt, der zu dem zweiten Thread gehört, den zweiten Schritt, der zu dem dritten Thread gehört etc. ausführen. Threads können hinzugefügt oder entfernt werden, ohne daß das Auswirkungen auf andere Threads hat, die ausgeführt werden. Threads können mit anderen Threads unter Verwendung von Benachrichtigungen oder Mitteilungen zwischen den Threads kommunizieren. Auch können Daten, die von verschiedenen. Threads gespeichert und verwendet werden, für einen Thread lokal sein, das heißt nur dem Thread bekannt oder durch den Thread zugreifbar sein, welcher die Daten geschaffen oder gespeichert hat.

Fig. 7 zeigt ein Blockdiagramm eines Konfigurationsdatenbankkommunikationssystems **200**, das Abonnentenbeziehungen zwischen mehreren Clients umsetzt. Eine Konfigurationsdatenbank enthält einen Datenbankserver **202**, der zwischen einer Konfigurationsdatenbank **203** und mehreren Clients **206** bis **208** in Kommunikationsverbindung steht. Die Konfigurationsdatenbank **203** ist so dargestellt, daß sie zwei Datenspeicher **210** und **212** enthält, von welchen jeder eine Konfigurationskomponente speichert, die durch einen oder mehrere der Clients **206** bis **208** abonniert ist. Während die Datenbank **203** so dargestellt ist, daß sie zwei Datenspeicher **210** und **212** hat, versteht sich, daß die Datenbank **203** mehrere oder andere Datenspeicher enthalten kann, wobei zur besseren Verständlichkeit der Darstellungen nur zwei gezeigt sind. Ferner versteht es sich, daß jede Anzahl von Clients unter der Benutzung des Servers **202** auf die Datenbank **203** zugreifen kann. Auch kann jeder Datenspeicher **210** und **212** zu einer anderen physischen Datenbank gehören, so daß ein Server verwendet werden kann, um Zugriff auf mehr als eine Datenbank zu gewähren.

Die Clients **206** bis **208** kommunizieren mit dem Server **202** über Kommunikationsverbindungen **213**, bei denen es sich um direkte Hochgeschwindigkeitsverbindungen wie z. B. eine Ethernetverbindung handeln kann oder welche langsame, mit niedriger Bandbreite arbeitende oder nicht permanente Verbindungen sein können, wie z. B. Satellitenverbindungen, zelluläre Verbindungen, Modemverbindungen etc. Jede erste Anforderung einer Konfigurationskomponente durch einen der Clients **206** bis **208** erstellt einen Clientthread innerhalb des Servers **202** und dieser Clientthread wird verwendet, um den Clients **206** bis **208** die angeforderten Komponenten aus dem entsprechenden Datenspeicher **210** und/oder **212** unter Verwendung eines gemeinsam genutzten Cache **214** innerhalb des Servers **202** zur Verfügung zu stellen, wie weiter unten detailliert beschrieben wird. Ein Clientthread kann auch verwendet werden, um Veränderungen in die Datenspeicher **210** und **212** zu schreiben, Komponentendaten hinzuzufügen oder Komponentendaten aus der Datenbank **203** zu löschen und eine Verriegelung von Komponentendaten innerhalb der Datenspeicher **210** und **212** auszuführen.

Allgemein ausgedrückt errichtet der gemeinsam genutzte Cache **214** ein Datenspeicherobjekt, das als Lightweight bezeichnet wird, für jedes der verschiedenen Datenelemente innerhalb der Datenbank **203**, welche zu einem bestimmten Zeitpunkt durch einen der Clients **206** bis **208** abonniert werden. Nur ein Lightweight muß für jede Konfigurationskomponente innerhalb der Datenbank **203** geschaffen werden und die Clientthreads für alle Clients, die diese Konfigurationskomponente abonnieren, greifen auf dasselbe Lightweight zu. Somit wird für das System von **Fig.** 7 ein Lightweight innerhalb des gemeinsam genutzten Cache **214** für jede der Konfigurationskomponenten innerhalb der Datenspeicher **210** bis **212** geschaffen, wenn wenigstens einer der Clients **206** und **208** die Komponente in dem Datenspeicher **210** abonniert und mindestens einer der Clients **206** bis **208** die Komponente in dem Datenspeicher **212** abonniert. Wenn jeder der Clients **206** bis **208** verschiedene Konfigurationskomponenten abonniert, enthalten die Threads für jeden Client verschiedene Lightweights, die mit verschiedenen Datenspeichern innerhalb der Konfigurationsdatenbank **203** verbunden sind. Wenn jedoch mehr als ein Client dieselbe Komponente innerhalb der Datenbank **203** abonniert, verwenden die Threads für diese Clients dieselben Lightweights innerhalb des gemeinsam genutzten Cache **114**. Aus diesem Grunde sind die Lightweights innerhalb des gemeinsam genutzten Cache nicht an einen Thread gebunden.

Allgemein wird jedesmal dann, wenn ein erster Client zuerst eine Komponente innerhalb der Datenbank **203** abonniert, ein Lightweight für diese Konfigurationskomponente innerhalb des gemeinsam genutzten Cache **214** geschaffen und dieses Lightweight liest und speichert eine Kopie der Konfigurationskomponente aus der Datenbank **203**. Anschließend wird bei jedem anderen Client, der dieselbe Konfigurationskomponente abonniert, dessen Clientthread mit demselben Lightweight verbunden, was bedeutet, daß der zweite, dritte, etc. Client, der diese Konfigurationskomponente abonniert, die Komponente aus dem Lightweight anstatt aus der Konfigurationsdatenbank **203** lesen kann, was die Anzahl der Lesevorgänge aus der Konfigurationsdatenbank **203** reduziert.

Der in **Fig.** 7 gezeigte Datenbankserver **202** enthält drei Clientthreads **216**, **217** und **218**, welche verschiedene Lese-/Schreibaktivitäten für die Clients **206**, **207** und **208** jeweils durchführen, und zwei Mitteilungsthreads **220** und **222**, welche die Benachrichtigungsaktivitäten bezüglich der Clientthreads **216**, **217** und **218** ausführen. Allgemein ausgedrückt benutzt jedes Clientthread **216**, **217** und **218** Serverkomponentenobjekte **226**, **227** und **228** jeweils zur Kommunikation mit den Clients durch die Kommunikationsverbindungen **230**. Die Serverkomponentenobjekte innerhalb eines bestimmten Clientthread kommunizieren mit verschiedenen Lightweights innerhalb des gemeinsam genutzten Cache **214** für jede verschiedene Komponente, auf die durch den Client zugegriffen wird, und führen diese Kommunikation unter Verwendung eines Komponentendatenwrappers, der zu jedem derartigen Lightweight gehört. Jeder Komponentendatenwrapper steuert den Zugriff zu einem

zugehörigen Lightweight und ist durch mehr als einen Clientthread zugreifbar.

In dem System von **Fig.** 7 abonniert der Client **216** die Komponenten innerhalb der Datenspeicher **210** und **212**, der Client **207** abonniert die Komponente innerhalb des Datenspeichers **210** und der Client **208** abonniert die Komponente innerhalb des Datenspeichers **212**. Wie **Fig.** 7 zeigt, verwendet der Clientthread **216** für den Client **206** eines der Serverkomponentenobjekte **226** zur Kommunikation mit einem ersten Komponentendatenwrapper **232**, welcher wiederum mit einem Lightweight **234** verbunden ist, der mit dem Datenspeicher **210** kommuniziert. Nach dem Einrichten kopiert das Lightweight **234** die Daten innerhalb des Datenspeichers **210** und macht diese Daten für jeden Client, der diese Daten abonniert, zugänglich. Eines der Serverkomponentenobjekte **226** des Clientthreads **216** steht auch mit einem Komponentendatenwrapper **240** in Kommunikation, welcher wiederum mit einem Lightweight **242** verbunden ist, das mit dem Datenspeicher **212** kommuniziert. Entsprechend kopiert nach dem Einrichten das Lightweight **242** die Daten aus dem Datenspeicher **212** und macht diese Daten für jeden Client, der diese Daten abonniert, zugänglich.

Der Clientthread **217** für den Client **207** verwendet das Serverkomponentenobjekt **227** zur Kommunikation mit dem Komponentendatenwrapper **232** und somit mit dem Lightweight **234**, um dadurch auf die Daten innerhalb des Lightweights **234** zuzugreifen, das heißt die Daten in dem Datenspeicher **210**. Andererseits verwendet der Clientthread **218** für den Client **208** die Serverkomponente **228** zur Kommunikation mit dem Komponentendatenwrapper **240**, um auf die Daten in dem Lightweight **242** zuzugreifen, die zu dem Datenspeicher **212** gehören. Somit abonnieren in dieser Konfiguration die Clients **206** und **207** dieselbe Konfigurationskomponente und als Resultat haben beide Clients **206** und **207** Zugriff auf das Lightweight **234** für diese Komponente. Entsprechend abonnieren die Clients **206** und **208** dieselbe Konfigurationskomponente und haben beide Zugriff auf das Lightweight **242** für diese Komponente. Als Resultat werden die Lightweights **234** und **242** jeweils von zwei Clientthreads gemeinsam genutzt und ermöglichen die gemeinsame Nutzung von Daten innerhalb des gemeinsam genutzten Cache **214** durch verschiedene Clients.

Wie noch detaillierter erläutert wird, erstellt der Server **202** die Serverkomponentenobjekte, die Komponentendatenwrapperobjekte und die Lightweights, um einen Clientthread zu schaffen oder zu erweitern, wenn ein Client eine Konfigurationskomponente innerhalb der Datenbank **203** abonniert. Nach der Erstellung werden die Clientthreads **216**, **217** und **218** von den Clients verwendet, um Lesevorgänge aus und Schreibvorgänge in die Serverkomponentenobjekte **226**, **227** und **228** durchzuführen, welche wiederum Lesevorgänge in dem bzw. aus dem gemeinsam genutzten Cache **214** über die Komponentendatenwrapper und die darin geschaffenen Lightweights durchführen. Der gemeinsam genutzte Cache **214** führt Lesevorgänge aus der und Schreibvorgänge in die Konfigurationsdatenbank **203** nach Erfordernis durch. Während für jeden Client ein anderer Clientthread geschaffen wird, kann jeder Clientthread ein oder mehrere Abonnements von Komponentendaten für einen Client handhaben. Allgemein wird für jedes Abonnement einer Komponente durch einen Client ein anderes Serverkomponentenobjekt geschaffen.

Der Mitteilungsthread **220** erfaßt Veränderungen, die in der Datenbank **203** erfolgt sind, und erfaßt insbesondere Veränderungen, die an den Datenspeichern **210** und **212** durchgeführt wurden, für welche Lightweights geschaffen wurden, indem der Status einer Veränderungsliste **244** geprüft wird, die innerhalb der Datenbank **203** geführt wird. Wenn eine Veränderung durchgeführt wurde, veranlaßt der Mitteilungsthread **220**, daß eine Mitteilung an jeden der Clientthreads gesendet wird, welche die veränderte Komponente, an der die Veränderung aufgetreten ist, abonnieren. Jeder dieser Clientthreads kann anschließend einen Client, der das entsprechende Serverkomponentenobjekt für diesen Clientthread benutzt, benachrichtigen. Die Benachrichtungsmitteilung, die dem Client gesendet wird, kann die neuen Komponentendaten enthalten oder den Client einfach benachrichtigen, daß neue Daten existieren und daß der Client einen manuellen Abhol- oder Lesevorgang dieser Daten über den Clientthread, der für die veränderte Konfigurationskomponente geschaffen wurde, durchführen muß. In ähnlicher Weise erfaßt der Laufzeitthread **222** bestimmte, an dem Prozeßsteuersystem durchgeführte Veränderungen, die von einem beliebigen Laufzeitserver **246** durchgeführt wurden, und benachrichtigt die Clientthreads, die Komponenten abonnieren, welche mit den veränderten Daten in Verbindung stehen, über die Veränderung.

Allgemein ausgedrückt wird der Zugriff auf Daten innerhalb der Konfigurationsdatenbank **203** auf der Basis Komponente per Komponente durchgeführt. Jede Komponente kann ein einzelnes Element innerhalb einer Konfigurationsdatenbank sein, wie beispielsweise ein Bereich, eine Steuereinrichtung, eine Einrichtung, ein Steuermodul etc., oder kann wenn gewünscht ein Baum von Elementen sein. Ferner versteht sich, daß jede Komponente eine übergeordnete bzw. Parent-Komponente sein kann und eine oder mehrere untergeordnete bzw. Chhild-Komponenten haben kann, die zu dieser gehören, und/oder jede Komponente eine untergeordnete Komponente einer übergeordneten Komponente sein kann. So kann beispielsweise ein Bereich eine übergeordnete Komponente mit einer Anzahl von untergeordneten Komponenten sein, die bestimmte Bereiche angeben. Jeder bestimmte Bereich kann eine übergeordnete Komponente mit untergeordneten Komponenten sein, welche die Steuereinrichtungen, die Benutzerschnittstellen, Steuermodule etc. innerhalb eines bestimmten Bereichs bezeichnen.

Fig. 8 zeigt einen Objektplan für die Objekte, die verwendet werden, um das Kommunikationssystem von **Fig.** 7 mit dem gemeinsam genutzten Cache umzusetzen, Im einzelnen hat ein Client **300**, bei dem es sich um einen beliebigen Client handeln kann, ein Clientkomponentenobjekt **301**, das beispielsweise jeder Teil von Konfigurationsdaten sein kann, auf den durch den Client für einen beliebigen Zweck zugegriffen wird bzw. der abonniert ist, beispielsweise zur Darstellung durch einen Benutzer in einer Konfigurationshierarchie wie der in **Fig.** 2 gezeigten. Die Clientkomponente **301** kann ein übergeordnetes Objekt bzw. ein Parentobjekt sein, zu dem ein oder mehrere untergeordnete Komponenten bzw. Childrenkomponenten gehören, wie durch den Rückkopplungspfad an der Clientkomponente **301**, der einen Punkt auf der Seite der untergeordneten Komponenten hat und das Fehlen eines Punkten auf der Seite der übergeordneten Komponente zeigt. In **Fig.** 8 bezeichnet die Verwendung eines Punktes eine Eins-zu-eins-Beziehung bzw. eine Eins-zu-mehreren-Beziehung, während das Fehlen eines Punktes exakt eine Eins-zu-eins-Beziehung bezeichnet. Wie bei dem Clientkomponentenobjekt **301** dargestellt, kann jede übergeordnete Clientkomponente eine oder mehrere untergeordnete oder untergeordnete Clientkomponenten haben, während jede untergeordnete oder untergeordnete Clientkomponente nur eine direkte übergeordnete Clientkomponente haben kann.

Jede Clientkomponente **301** ist mit einem Smart Proxy **302** verbunden, der zu dieser Clientkomponente an dem Client **300** zugehörig ist. Der Smart Proxy **302** ist für die Abwicklung der Kommunikation für die Clientkomponente

301 über eine Kommunikationsverbindung 304 zuständig. Insbesondere ist der Smart Proxy ein Verteilobjekt, das Clientanfragen empfängt, die eine Komponente betreffen, über die Kommunikationsverbindung einen Server 306 anruft, um diese Anfragen dem Server 306 mitzuteilen, Aktualisierungen empfängt, im Cache gespeicherte Daten und Mitteilungen, die von dem Server 306 gesendet werden, speichert, und diese Daten an das Clientkomponentenobjekt 301 weiterleitet. Selbstverständlich führt der Smart Proxy 302 die Operationen aus, die erforderlich sind, um Kommunikation über die Kommunikationsverbindung 304 durchzuführen, in jeder gewünschten Weise unter Verwendung jeder gewünschten Kommunikationstechnik, wie beispielsweise unter Verwendung der verzögerten Bestätigungsroutine, die in der US-Patentanmeldung mit dem Titel "Deferred Acknowledgment Communications and Alarm Management" beschrieben ist, die vorstehend angeführt wurde. Die Kommunikationsverbindung 304 kann jede gewünschte Kommunikationsverbindung sein, darunter beispielsweise eine langsame oder mit niedriger Bandbreite arbeitende Verbindung, beispielsweise eine Satelliten-, zelluläre-, Internet- oder andere gemeinsam genutzte Fernnetzwerkverbindung, oder könnte eine nicht permanente Verbindung sein, wie beispielsweise eine Modemverbindung, oder könnte eine direkte Verbindung oder Hochgeschwindigkeitsverbindung, wie etwa eine dezidierte Ethernet-Verbindung sein, falls dies erwünscht ist.

Der Server 306 enthält Serverkomponentenobjekte 308, die für die Kommunikation mit dem Smart Proxy 302 eines Client über die Kommunikationsverbindung 304 verantwortlich sind. Jede Serverkomponente 308 ist für die Aufzeichnung und das Halten mindestens eines Teils der Konfigurationshierarchie verantwortlich, auf die von einem bestimmten Client zugegriffen oder abonniert wird, um eine Schnittstelle zu bieten, um auf Eigenschaften einer Konfigurationskomponente innerhalb einer Konfigurationsdatenbank zuzugreifen, und zum Empfangen von Aktualisierungsereignissen und Weiterleiten dieser Ereignisse zu dem Smart Proxy 302 für einen Client. Wie Fig. 8 zeigt, kann eine Serverkomponente 308 mit anderen Serverkomponenten durch eine Parent-Child-Beziehung, das heißt eine Beziehung eines übergeordneten Elements zu einem untergeordneten Element, verbunden sein, und diese Parent-Child-Beziehung reflektiert die Parent-Child-Beziehung, die für die zugehörigen Clientkomponenten 301 geschaffen wurde. Insbesondere gibt diese Beziehung die Clienthierarchie, d. h. die Konfigurationshierarchie, welche der Client abonniert hat, wieder. Jede Serverkomponente 308 kann mit genau einem Komponentendatenwrapper 310 verbunden sein und an diesem interessiert sein, bei dem es sich um ein Objekt handelt, welches innerhalb des gemeinsam genutzten Cacheabschnittes des Servers 306 errichtet wird. Wie von Fig. 8 angegeben, kann jedoch ein Komponentendatenwrapper 310 mit vielen Serverkomponenten für verschiedene Clients oder Clientthreads verbunden sein.

Der Komponentendatenwrapper 310 ist eine Schnittstelle zu gemeinsam genutzten Daten, das heißt Daten, die in dem gemeinsam genutzten Cache 214 von Fig. 7 von mehreren Clients oder Clientthreads gemeinsam genutzt werden. Die Komponentendatenwrapper 310 können Hierarchiebeziehungen aufzeichnen (und sind ein übergeordneter Satz (Superset) von Clienthierarchien, die von allen abonnierenden Clients betrachtet werden), können verweisende Serverkomponenten registrieren, können Serverkomponenten benachrichtigen, wenn Veränderungsvorgänge erfaßt werden, können das Datum der letzten Modifikation aufzeichnen (das verwendet werden kann, um Ereignisse von der Konfigurationsdatenbank zu akzeptieren oder zurückzuweisen) und können eine Verriegelung von gemeinsam genutzten Daten bewirken. Ähnlich wie die Serverkomponente 308 kann der Komponentendatenwrapper 310 mit anderen Komponentendatenwrappern in einer Parent-Child-Beziehung stehen bzw. mit diesen verbunden sein, wobei diese Beziehung die Beziehung wiedergibt, die für alle Serverkomponenten 308 und alle Clientkomponenten 301 geschaffen wird. Es versteht sich, daß jeder Komponentendatenwrapper 310 mit zahlreichen Serverkomponenten 308 verbunden sein kann, da alle Clientthreads, die auf eine bestimmte Konfigurationskomponente zugreifen, durch denselben Komponentendatenwrapper 310 gehen. Der Komponentendatenwrapper 310, der von einem Komponentendatenwrappergenerator 312 geschaffen wird, wenn ein erster Client erstmals ein Element innerhalb der Konfigurationsdatenbank abonniert, verfolgt allgemein, welche Serverkomponenten 308 bestimmte Komponenten innerhalb der Konfigurationsdatenbank abonnieren und arbeitet so, daß er jede der abonnierenden Serverkomponenten 308 über Veränderungen dieser Daten benachrichtigt.

Jeder Komponentendatenwrapper 310 ist mit höchstens einem Lightweightobjekt 314 verbunden bzw. enthält ein solches, das eine Kopie der Komponentendaten in der Konfigurationsdatenbank speichert, welche eine oder mehrere Serverkomponenten 308 abonnieren. Mit anderen Worten wird ein Komponentendatenwrapper 310 zwischen eine oder mehrere Serverkomponenten 308 und ein einzelnes Lightweight 314 eingefügt, um den Zugriff auf das Lightweight 314 durch mehrere Serverkomponenten 308 abzuwickeln. Allgemein ausgedrückt ist das Lightweight 314 ein Aufbewahrungsort für Daten, die von den Serverkomponenten 308 gemeinsam genutzt werden, und wird verwendet, um sich mit dem Mitteilungsthread 220 von Fig. 7 für Aktualisierungsbenachrichtigungen abzustimmen, welche Veränderungen betreffen, die an der Konfigurationsdatenbank 203 von Fig. 7 durchgeführt werden.

Der Komponentendatenwrappergenerator 312 ist verantwortlich für das Erhalten/Schaffen von Komponentendatenwrappern 310 und Lightweights 314. Wenn ein Client angibt, daß er eine Komponente in der Konfigurationsdatenbank 203 abonnieren will, prüft der Komponentendatenwrappergenerator 312 zunächst, ob ein Komponentendatenwrapper 310 für diese Komponente bereits geschaffen wurde. Wenn dies der Fall ist, verbindet der Komponentendatenwrappergenerator 312 den Clientthread mit dem Komponentendatenwrapper 310 für diese Komponente. Wenn dies nicht der Fall ist, schafft der Komponentendatenwrappergenerator 312 einen neuen Komponentendatenwrapper 310 und ein Lightweight 314, welche veranlassen, daß die angeforderte Information von der Konfigurationsdatenbank 203 in das Lightweight 314 geladen wird, und schafft oder erweitert den Thread für den Client, der diesen neuen Komponentendatenwrapper 310 und das zugehörige Lightweight 314 nutzt. Wenn der Komponentendatenwrappergenerator 312 einen neuen Wrapper 310 schafft, weist der Komponentendatenwrappergenerator 312 dem Wrapper 310 ein einzigartiges unveränderliches Kennzeichen zu. Dieses Kennzeichen wird zu den verweisenden Serverkomponenten 308 und ihren entsprechenden Clientkomponenten 301 weitergeleitet, was es erlaubt, eine Gruppe von Komponenten oder einzelne Komponenten eindeutig zu identifizieren, was wiederum viele schwierige Umbenennungsprobleme vermeidet und eine effizientere Überwachung einer Liste von Identifikationen über eine langsame Kommunikationsverbindung bietet, wenn beispielsweise ein gesamter Baum oder Zweig eines Baumes von einem Client angefordert wird.

Ein Verriegelungsmanager 316 ermöglicht es jedem Komponentendatenwrapper 310, den Zugriff auf seine zu-

gehörigen Lightweights **314** durch verschiedene Clients zu steuern. Das Verriegeln eines Lightweights **314** kann das Verriegeln eines oder mehrerer anderer Lightweights verursachen, so daß beispielsweise das Verriegeln eines Lightweights, das übergeordnete Kompontendaten speichert, automatisch die Verriegelung von Lightweights verursacht, die untergeordnete Komponenten für diese übergeordnete Komponente enthalten. In ähnlicher Weise kann jedes Lightweight **314** automatisch bei Verriegelung eines anderen Lightweights verriegelt werden, so daß ein Lightweight, das eine untergeordnete Komponente enthält, verriegelt werden kann, wenn das Lightweight, das die übergeordnete Komponente für diese untergeordnete Komponente enthält, verriegelt wird. Auf diese Weise kann der Verriegelungsmanager **316** veranlassen, daß ein Komponentendatenwrapper **310** sein zugehöriges Lightweight **314** verriegelt, wenn ein Lesevorgang aus oder ein Schreibvorgang in dieses Lightweight innerhalb eines bestimmten Clientthreads durchgeführt wird und die Verriegelung dieses Lightweights kann automatisch die Verriegelung von anderen Lightweights verursachen, wie etwa Lightweights, die untergeordnete Komponenten enthalten, die sich auf das Lightweight beziehen, das verriegelt wurde.

Ferner verwendet jedes Lightweight **314** eines oder mehrere transaktionale Speicherobjekte (XMs) **318**, um Komponenteneigenschaften aus der bzw. in die Konfigurationsdatenbank **203** von **Fig. 7** unter Verwendung eines Kontextspeichers zu lesen oder zu schreiben. Ein XM **318** wird jedesmal dann gestartet, wenn ein Lightweight **314** einen Lesevorgang oder einen Schreibvorgang an der Konfigurationsdatenbank **203** durchführen muß. Das XM **318** schafft einen Zeiger in den Kontextspeicher (der lokal für den Clientthread ist), der verwendet wird, um diesen lesen oder schreiben zu lassen, und das XM **318** veranlaßt den Lesevorgang aus der oder den Schreibvorgang in die Konfigurationsdatenbank **203** unter Verwendung des Kontextspeichers. Das XM **318** kann ferner verwendet werden, um Datenbankobjekte innerhalb der Datenbank **203** zu schaffen/zu löschen und die Registrierung eines Benachrichtigungsinteresses in der Konfigurationsdatenbank **203** zu entfernen. Sobald der entsprechende Lesevorgang oder Schreibvorgang für ein Lightweight **314** durchgeführt wurde, wird das XM **318** freigegeben. Die in dem Kontextspeicher gespeicherten Daten sind jedoch noch vorhanden und der Zeiger zu diesem Speicher wird in dem lokalen Threadsspeicher gespeichert, um bei Bedarf später verwendet zu werden. Die Verwendung von XMs ist ein Standardvorgang in Objektdatenbankmanagern und wird daher hier nicht weiter erläutert.

Eine Benachrichtigungsmaschine **320** stimmt sich mit jedem XM **318** ab und erfaßt Änderungen an jedem der Datenbankwerte innerhalb der Datenspeicherorte in der Konfigurationsdatenbank **203**, auf die von den XMs **318** verwiesen wird. Die dortigen Veränderungen können beispielsweise durch einen Client verursacht sein, der einen Schreibvorgang in der Konfigurationsdatenbank **203** verursacht, einen Laufzeitprozeß oder einen anderen externen Prozess. Wenn die Benachrichtungsmaschine **320** eine Veränderung an den Datenbankelementen erfaßt, auf die von einem der XMs **318** verwiesen wird, benachrichtigt die Benachrichtigungsmaschine **320** den Komponentendatenwrapper **310**, welcher das Lightweight **314** aufweist, das zu diesem XM **318** gehört, welches wiederum alle Clients benachrichtigt, welche die Komponente abonniert haben, daß eine Veränderung durchgeführt wurde und daß neue Werte für die Komponente verfügbar sind. Um diese Operation auszuführen, verwendet die Benachrichtigungsmaschine **320** ein Komponentendatenregister (auch als ein Wrapperplan bezeichnet) **322**, bei dem es sich um einen Plan handelt, in dem jeder Komponentendatenwrapper **310** mit einem Datenbankort verzeichnet ist, auf den von einem XM **318** verwiesen wird. Genauer ausgedrückt ist das Komponentendatenregister **322** eine Nachschlagtabelle, die Komponentendatenwrapper **310** mit Benachrichtungscookies oder Identifikationskennzeichen in Verbindung setzt, die als ein Resultat der Operation der XMs **318**, welche auf die Datenbank **203** zugreifen, geschaffen werden. Gelegentlich werden diese Identifikationskennzeichen auch als die "magischen Zahlen" bezeichnet, und allgemein ausgedrückt bezeichnen diese Identifikationskennzeichen einen Ort oder identifizieren eine Komponente, die in der Datenbank **203** gespeichert ist. Die Benachrichtigungsmaschine **320** erhält eine Liste von Aktualisierungsereignissen, die mit jedem der Datenbankorte in Beziehung stehen, die den magischen Zahlen zugehörig sind, und verwendet das Komponentendatenregister **322**, um die Komponentendatenwrapper **310** zu finden, die jeder Komponente zugeordnet sind, für welche eine Veränderung aufgetreten ist. Als Resultat ist die Benachrichtigungsmaschine **320** ein Ereignisgenerator, der periodisch eine Veränderungsliste von der Konfigurationsdatenbank **203** erhält und Veränderungsereignisse an die Komponentendatenwrapper **310** abgibt, für welche Veränderungen aufgetreten sind.

Allgemein ausgedrückt registriert jeder Komponentendatenwrapper **310** alle XMs, die zu seinem Lightweight gehören, in dem Komponentendatenregister **322**, wenn Clientthreads geschaffen werden, um dadurch Veränderungsbenachrichtigungen zu erhalten. Im Gegenzug erhalten die Datenkomponentenwrapper **310** ein einzigartiges Cookie für jede Registrierung und diese Cookies werden in dem Komponentendatenregister bzw. dem Wrapperplan **322** gespeichert. Ansprechend auf eine Veränderungsbenachrichtigung zwingt ein Komponentendatenwrapper **310** sein zugehöriges Lightweight **314**, einen neuen Ladevorgang von der Datenbank durchzuführen, sofern erforderlich, und die Konfigurationshierarchie zu aktualisieren. Ferner sendet der Komponentendatenwrapper **310** Aktualisierungsmitteilungen zurück zu jeder der Serverkomponenten **308**, die mit dem Komponentendatenwrapper **310** in Beziehung stehen und welche auf diesen zugreifen, und die Serverkomponenten **308** werden anschließend mit dem gemeinsam genutzten Cache, das heißt den Lightweights **314**, neu synchronisiert. Schließlich senden die Serverkomponenten **308** eine Aktualisierungsmitteilung an die Clientskomponenten **301**, welche eine Neusynchronisierung mit den Serverkomponenten **308** durchführen.

In **Fig. 9** ist der Status des Datenbankservers **306** (bei dem es sich um den Server **202** von **Fig. 7** oder jeden der Datenbankserver von **Fig. 1**, 3 bis 6 handeln kann) beim Startup dargestellt, das heißt bevor ein Client Daten in der Konfigurationsdatenbank **325** (bei der es sich beispielsweise um die Datenbank **203** von **Fig. 7** oder jede der Konfigurationsdatenbanken von **Fig. 1** und 3 bis 6 handeln kann) abonniert hat. Der Server **306** enthält einen Pre-Start Thread **326**, der keine Serverkomponentenobjekte enthält, einen gemeinsam genutzten Cache **327**, der keine Komponentendatenwrapper oder Lightweights hat, und einen leeren Datenbankkontextspeicher **328**. Der gemeinsam genutzte Cache **327** des Servers **306** enthält einen Komponentendatenwrappergenerator **312** und einen Verriegelungsmanager **316**, der die Verriegelung von Komponenten, die aus der Datenbank **325** ausgelesen werden, koordiniert. Wie **Fig. 9** zeigt, enthält ein Benachrichtigungsthread eine Benachrichtigungsmaschine **320**, die Veränderungen erfaßt, die an Daten in der Datenbank **325** wie vorstehend beschrieben durchgeführt wurden, und ein Komponentendatenregister oder einen Wrapperplan **322**, der Datenkomponentenwrapper in dem gemeinsam genutzten Dache **327** mit Orten in der Konfigurationsdaten-

bank **325** aufzeichnet. Da in dem gemeinsam genutzten Cache **327** keine Datenwrapper existieren, ist der Wrapperplan **322** selbstverständlich leer.

Entsprechend enthält ein Laufzeitthread eine Benachrichtigungsmaschine **336**, die Veränderungen erfaßt, die an bestimmten Teilen der Konfigurationsdaten von Laufzeitservices **338** durchgeführt wurden, bei welchen es sich um beliebige Services handeln kann, wie z. B. automatische Erfassungsservices oder andere Anwendungen, die während der Laufzeit oder während der Konfigurationsaktivitäten ablaufen, sowie einen Plan **340**, der es ermöglicht, daß die Laufzeitbenachrichtigungsmaschine **336** Veränderungsmitteilungen an entsprechende Komponentendatenwrapper in den Clientthreads sendet, die in dem Server **306** ablaufen. Beim Startup erzeugt die Laufzeitbenachrichtigungsmaschine **336** eine Instanz eines Objektes, die als Erfassung abläuft, um bestimmte Veränderungen in der Konfiguration des Prozeßsteuernetzwerks zu erfassen, wie z. B. das Hinzufügen eines neuen Knotens oder das Hinzufügen von neuen Einrichtungen (wie etwa außer Dienst gestellte Steuereinrichtungen) innerhalb des Systems. In einer Ausführungsform fragt die Benachrichtigungsmaschine **336** die Laufzeitservices **338**, bei welchen es sich um beliebige Laufzeitservices oder Anwendungen handeln kann, nach Listen von außer Dienst gestellten Steuereinrichtungen und Fieldbuseinrichtungen ab. Der Plan **340** verweist allgemein auf einen Speicherort oder einen Speicherbereich, wo entsprechende Datenwrapper (nach deren Erstellung) registriert sind, um Informationen zu empfangen, die zu der Erstellung eines neuen Knotens, einer neuen Einrichtung etc. innerhalb des Prozeßsteuersystems gehören. Selbstverständlich müssen nicht alle Datenwrapper bei dem Laufzeitthread registriert werden, sondern nur diejenigen Wrapper, die zu Komponenten gehören, welche sich auf die neu hinzugefügten Einrichtungen beziehen, wie etwa die in **Fig.** 2 dargestellte Komponente der außer Dienst gestellten Steuereinrichtung. Beim Startup verweist der Plan **340** auf einen Speicherort, an dem keine Wrapper registriert sind.

In **Fig.** 10 sind die Operationen des Servers **306** dargestellt, wenn ein erster Client **342** Verbindung aufnimmt und eine Komponente in der Konfigurationsdatenbank **325** abonniert. Im einzelnen läuft die Clientanwendung an, nimmt Verbindung mit dem Server **306** über eine Kommunikationsverbindung **344** unter Verwendung eines Smart Proxy **346** auf, um eine Komponente **348** innerhalb der Konfigurationshierarchie anzufordern. Zu diesem Zeitpunkt weist der Server **306** den Pre-Start-Clientthread **326** dem Client **342** zu und erstellt einen neuen Pre-Start-Clientthread (nicht dargestellt), der von dem nächsten Client zu verwenden ist, der Verbindung mit dem Server **306** aufnimmt. Typischerweise beginnt der Client **342** mit der Anforderung eines Wurzelobjektes, wie etwa der Bereiche-Komponente. An diesem Punkt wird in dem Server **306** eine Bereiche-Serverkomponente **350** geschaffen und die Bereiche-Serverkomponente **350** fordert die Bereiche-Komponente von dem Wrappergenerator **312** an. Der Wrappergenerator **312** erkennt, daß kein Komponentendatenwrapper (nachfolgend als "Wrapper" bezeichnet) für die Bereiche-Komponente in dem gemeinsam genutzten Cache **327** geschaffen wurde und erstellt einen derartigen Wrapper **354** (einen Bereiche-Wrapper), wobei der Wrapper **354** der Bereiche-Serverkomponente **350** innerhalb des Clientthread **326** zugeordnet wird. Der Wrappergenerator **312** erstellt ein Lightweight **360**, das alleine dem neu erstellten Wrapper **354** zugeordnet ist und das mit dem Datenbank-"Standort"-KContext **362** für den Clientthread verbunden ist. Der Standort-Kontext wird verwendet, da die Bereiche-Wurzelkomponente angefordert wurde und diese Komponente mit einem Standortobjekt in der Datenbank **325** im Zusammenhang steht. Zu dieser Zeit führt das Lightweight **360** einen Aufruf in der Datenbank **325** betreffend die Bereiche-Komponente durch, indem ein XM (nicht dargestellt) erstellt wird. Das XM verweist auf einen Speicherort in dem Kontextspeicher **328**, der von dem Clientthread **326** zu verwenden ist, und veranlaßt den Datenbankmanager, die Bereiche-Komponente auszulesen und die Komponentendaten in dem Standort-Kontextspeicher **362** zu speichern. Anschließend werden die Daten in das Lightweight **360** geladen und das XM wird freigegeben.

Während dieses Prozesses registrieren der Wrappergenerator **212** oder das XM den neugeschaffenen Bereiche-Wrapper **354** in dem Wrapperplan **325** zur Benachrichtigung über Veränderungen, wenn Veränderungen in dem Gegenstück in der Datenbank auftreten, in diesem Fall in der Bereiche-Komponente der Datenbank **325**. Diese Registrierung ist durch die Linie **363** dargestellt. Um diese Registrierung durchzuführen, wird ein eindeutiges Identifizierungskennzeichen (ID) für die Kombination Wrapper/Datenbankstandort geschaffen und dieses ID wird in eine Registrierungsmitteilung gesetzt, die zu dem Benachrichtigungsthread gesendet wird. Bei Verarbeitung der Registrierungsmitteilung fügt das Benachrichtigungsthread einen Eintrag in den Wrapperplan **322** für den Bereiche-Wrapper **354** ein, der durch das ID als Schlüssel gekennzeichnet ist. Dieser Eintrag verbindet die magische Zahl, die dem XM zugehörig ist, welches auf ein Objekt oder einen Ort in der Datenbank **325** verweist, mit dem Wrapper **354**.

Nachfolgend wird die Bereiche-Komponente von dem Wrapper **354** zu der Serverkomponente **350** gesendet, die mit den Daten der Bereiche-Komponente aktualisiert wird. Die Serverkomponente **350** sendet anschließend die Daten der Bereiche-Komponente an das Smart Proxy **346** des Client **342**, wo die Komponentendaten an die Clientkomponente **348** abgegeben werden.

Anschließend überwacht der Benachrichtigungsthread Veränderungen an der Bereiche-Komponente innerhalb der Datenbank **325**, und sucht wenn Veränderungen durchgeführt werden, das ID für den Wrapper, der zu dieser Komponente innerhalb der Datenbank **325** gemäß der Definition in dem Wrapperplan **322** gehört. Der Benachrichtigungsthread benachrichtigt anschließend den Wrapper, in diesem Fall den Wrapper **354**, über die Veränderung. Falls erforderlich, aktualisiert der Wrapper **354** das Lightweight **360**, so daß es die Veränderung wiedergibt, indem ein Auslesen der Datenbank **325** veranlaßt wird. Danach weist der Wrapper **354** die Serverkomponente **350** über die Veränderung an und die Serverkomponente **350** kann die neuen Daten für die Bereiche-Komponente abfragen, welche neuen Daten aus dem Lightweight **360** ausgelesen werden können. Danach benachrichtigt die Serverkomponente **350** den Client **342** über die Veränderung und der Client **342** kann die neuen Daten für die Bereiche-Komponente von der Serverkomponente **350** anfordern, so daß die Veränderung in der Bereiche-Komponente entweder automatisch oder durch Benachrichtigung des Clients über die Veränderung und Abwarten, bis der Client die neuen Komponentendaten abfragt, an die Clientkomponente **348** weitergegeben wird. Während der Client **342** ein manuelles Abziehen der neuen Komponentendaten aus der Serverkomponente **350** durchführen könnte, wird nachfolgend zum Zweck der Erörterung angenommen, daß neue Daten oder veränderte Daten automatisch zu einem Client gesendet werden, während eine Veränderung erfaßt wird.

In ähnlicher Weise verwendet der Laufzeitbenachrichtigungsthread die Benachrichtigungsmaschine **336**, um bestimmte Veränderungen zu erfassen, die in der Konfigura-

tion des Prozeßsteuersystems basierend auf der Operation der Laufzeitservices **338** durchgeführt wurden. Wenn eine bestimmte Veränderung aufgetreten ist, beispielsweise das Hinzufügen eines Knotens oder einer Einrichtung, sucht die Benachrichtigungsmaschine **336** den Speicherort, der für diese Veränderung von dem Plan **340** angegeben ist, um festzustellen, ob Wrapper eine Registrierung eingerichtet haben, um einen Hinweis auf die erfaßte Veränderung zu erhalten. Wenn dies der Fall ist, sendet die Benachrichtigungsmaschine **336** eine Ereignismitteilung an den bzw. die registrierten Wrapper, welche diese Mitteilungen in ähnlicher Weise verarbeiten können, in der Ereignismitteilungen verarbeitet werden, die von der Benachrichtigungsmaschine **320** erzeugt wurden. Es versteht sich, daß ein Datenwrapper, der Benachrichtigungen über Veränderungen erhalten will, die von dem Laufzeitbenachrichtigungsthread erfaßt wurden, eine Registrierung an dem Speicherort durchführt, der für diese Veränderung von dem Plan **340** angegeben ist, in dem beispielsweise das ID des Wrappers gespeichert wird, um die Veränderungsbenachrichtigung an dem Speicherort zu erhalten, wenn der Datenwrapper geschaffen wird.

Wie **Fig.** 11 zeigt, kann der Client **342** anschließend den Wunsch haben, untergeordnete Elemente der Bereiche-Clientkomponente **348** zu laden. In diesem Fall ruft der Client **342** einen Befehl zum Laden von untergeordneten Elementen und die Bereiche-Wurzelkomponente auf und dieser Befehl wird durch den Proxy **346** zu der Serverkomponente **350** und von dort zu dem Bereiche-Lightweight **360** weitergeleitet. Das Bereiche-Lightweight **360** fragt die Konfigurationsdatenbank **325** unter Verwendung des Kontextspeichers **362** für den Clientthread **326** ab, um Informationen über die untergeordneten Komponenten für Bereiche zu erhalten. Diese Daten werden zurückgegeben und ein Lightweight und ein zugehöriger Datenwrapper werden anschließend für jede untergeordnete Komponente geschaffen, die zu der Bereiche-Komponente gehört. In **Fig.** 11 ist die Bereiche-Komponente so dargestellt, daß sie nur eine untergeordnete Komponente enthält, die den Namen Area_A und die Beschreibung "Fred" trägt, und diese neuen untergeordneten Komponentendaten werden in einem neuen Lightweight **370** gespeichert. Andere untergeordnete Bereiche-Komponenten könnten auch zu der Bereiche-Komponente gehören und in diesem Fall würde für jedes dieser untergeordneten Elemente zusammen mit den anderen oben beschriebenen Objekten ein neues Lightweight geschaffen.

Bei dem Erstellen des neuen Lightweight **370** schließt der Bereichewrapper **354** (oder der Wrappergenerator **312**) das neue Bereiche-Lightweight **370** in einem separaten Wrapper **372** ein, der mit dem Bereiche-Wrapper **354** als ein untergeordneter Wrapper verbunden ist. Der Bereiche-Wrapper **354** führt die neue Bereiche-Komponente auch zu der Serverkomponente **350** zurück. Das neue Lightweight **370** erstellt ein XM, um eine Registrierung in der Datenbank **325** über den Kontextspeicher **376** einzurichten, um spezifische Daten zu erhalten, die zu der untergeordneten Bereich-Komponente gehören. Der Wrapper **372** oder das Lightweight **370** senden ferner eine Registrierungsmitteilung an den Benachrichtigungsthread mit einer Liste der neuen Wrapper-IDs (in diesem Fall ein Wrapper-ID) und der magischen Datenbankzahlen (in diesem Fall eine neue magische Zahl), die zu diesem gehören. Das neue ID wird in dem Wrapperplan gespeichert, um den neuen Wrapper **372** anzuzeigen, wie durch die Linie **378** dargestellt. Die magische Zahl, welche den Ort der neuen untergeordneten Bereich-Komponente in der Datenbank **325** angibt, wird ebenfalls an den Wrapperplan **322** abgegeben und für dieses ID gespeichert.

In der Zwischenzeit erzeugt die Bereiche-Serverkomponente **350** eine untergeordnete Bereich-Serverkomponente **380** für die neue untergeordnete Bereich-Komponente und bettet dabei einen Zeiger in der neuen untergeordneten Bereich-Serverkomponente **380** auf den Wrapper **372** ein, der für die untergeordnete Serverkomponente **380** geschaffen wurde. Die Serverkomponente **350** sendet ebenfalls eine Liste von untergeordneten Komponenten (in diesem Fall eine Liste mit einem Eintrag) über die Kommunikationsverbindung und den Proxy **346** an die Bereiche-Clientkomponente **348** und die Bereiche-Clientkomponente **348** erzeugt eine untergeordnete Bereich-Clientkomponente und bettet einen Serverkomponentenbezug (auf die Serverkomponente **380**) in der untergeordneten Bereich-Clientkomponente **382** ein, um die Kommunikation zwischen der neuen untergeordneten Client-Komponente **382** und der neuen untergeordneten Serverkomponente **380** zu ermöglichen. Ein Smart Proxy **384** für die Clientkomponente **382** wird anschließend geschaffen, was den Prozeß der Erweiterung des Clientthread **326**, um ein Abonnement für eine neue untergeordnete Komponente innerhalb der Datenbank **325** einzuschließen, vollendet. Selbstverständlich würden dieselben Vorgänge für andere untergeordnete Komponenten einer bestimmten Komponente stattfinden und der Clientthread **326** kann erweitert werden, um mehr als eine untergeordnete Komponente zu abonnieren, wenn mehr als eine untergeordnete Komponente bzw. mehr als ein Child vorhanden ist. Entsprechend kann der Clientthread **326** weiter erweitert werden, indem untergeordnete Komponenten erhalten werden, die zu der neu abonnierten untergeordneten Komponente **382** gehören oder indem eine weitere Wurzelkomponente angefordert wird.

In **Fig.** 12 ist der Prozeß, in dem ein zweiter Client **386** dieselben Komponenten abonniert, die durch den ersten Client **342** abonniert wurden, im Detail beschrieben. Hier sind die Benachrichtigungsthreads des Servers **306** aus Gründen der Übersichtlichkeit nicht gezeigt. Insbesondere nimmt der zweite Client **386** mit dem Server **306** in ähnlicher Weise wie der erste Client Verbindung auf und abonniert die Bereiche-Komponente. Nach Herstellung der Verbindung mit dem Server **306** wird der zweite Client **386** einem zweiten Pre-Start-Thread **387** zugewiesen. In diesem Beispiel fragt der Client **386** die Bereiche-Wurzelkomponente ab, welche eine Bereiche-Serverkomponente **388** in dem zweiten Clientthread erstellt. Die Serverkomponente **388** fragt den Wrappergenerator nach einem Wrapper für die Bereiche-Wurzelkomponente, und da der Wrapper **354** bereits in dem gemeinsam genutzten Cache **327** für die Bereiche-Wurzelkomponente existiert, sendet der Wrappergenerator **312** einen Zeiger auf den Wrapper **354** zurück. An diesem Punkt fragt die Bereiche-Serverkomponente **388** den Wrapper **354** nach den Bereiche-Wurzelkomponentendaten ab, die aus dem Lightweight **360** ausgelesen werden, das bereits für den ersten Client **342** geschaffen wurde. Die Bereich-Wurzelkomponentendaten, die in dem Lightweight **360** gespeichert sind, werden zu der Bereiche-Serverwurzelkomponente **388** zurückgeleitet und von dort an die Client-Bereiche-Komponente **389**.

Bei einer Anforderung für untergeordnete Elemente der Bereiche-Wurzelkomponente **389** durch den zweiten Client **386** fragt die Bereiche-Wurzelserverkomponente **388** den Wrappergenerator **312** nach untergeordneten Wrappern, und da diese Wrapper (in diesem Fall ein Wrapper) bereits vorhanden sind, wird der Zeiger für den Wrapper **372** von dem Wrappergenerator **312** an die Bereiche-Wurzelserverkomponente **388** abgegeben. Die Serverkomponente **388** erzeugt dann eine untergeordnete Bereich-Serverkomponente **390**, die einen Zeiger auf den Wrapper **372** hat. Die Bereiche-Wurzelserverkomponente **388** sendet die Information über das untergeordnete Element an den zweiten Client **386**, der

die geeignete Clientkomponente **392** erzeugt, und den Smart Proxy **393**, der mit der Serverkomponente **390** kommuniziert. Die untergeordnete Bereich-Serverkomponente **390** liest auch das Lightweight **370** über den Wrapper **372** und sendet diese Daten an die Clientkomponente **392** über den Proxy **393**. Da in diesem Prozeß keine Anrufe an die Datenbank erfolgten, ist der Datenbankkontext des zweiten Client leer, da alle Informationen durch die Lightweights **360** und **370** kamen, welche den Datenbankkontext des ersten Clientthread **326** nutzten, um diese Daten zu erhalten. Wenn jedoch der zweite Client **386** Daten anfordert, für die kein Lightweight erstellt wurde, würde der Wrappergenerator **312** einen Wrapper und ein Lightweight für diese Komponente erstellen und Verbindung mit der Datenbank **325** über den Datenbankkontext für den zweiten Clientthread **387** aufnehmen, um diese Daten von der Datenbank **325** zu erhalten.

Es versteht sich, daß jeder andere Client, der nun die Bereich-Wurzel oder die untergeordnete Bereich-Komponente abonniert, in der Lage ist, diese Daten aus dem gemeinsam genutzten Cache **327** zu erhalten, ohne daß Anrufe an die Datenbank **325** erforderlich sind, was die Belastung der Datenbank **325** verringert. Entsprechend benachrichtigt bei Erhalt einer Ereignisbenachrichtigung von einem Benachrichtigungsthread der Datenwrapper **354** die Serverkomponente **388** für den Client **386** sowie die Serverkomponente **350** für den ersten Client **342**, um beide Clients über die Veränderung zu benachrichtigen.

In **Fig.** 13 sind die Aktivitäten beschrieben, die mit dem Benachrichtigungsthread beim Erfassen einer Veränderung in einem Konfigurationsdatenelement und der Art und Weise, auf die diese Veränderungsbenachrichtigung automatisch zu jedem abonnierenden Client zurückgeführt wird, in Zusammenhang stehen. In diesem Fall verändert der zweite Client **386** den Namen des Area_A von "Fred" in "Wilma" und gibt eine Schreib- oder Übertragungsmitteilung mit dieser Veränderung an den Server **306** auf. Die Serverkomponente **390** lädt den neuen Namen und gibt diesen Namen an den Wrapper **372** ab, der den Namen in das Lightweight **370** lädt. Das Lightweight **370** lädt automatisch den neuen Namen in den Datenbankkontext **394** für den zweiten Client **386**. Von hier wird der neue Name in der Datenbank **325** gespeichert. Es ist jedoch nichts an der Version der Area_A-Komponente des ersten Client geschehen. In der Vergangenheit wäre es erforderlich gewesen, daß der erste Client **342** periodisch die Datenbank **325** nach Veränderungen abfragt, was zu einer Vielzahl von Lesevorgängen aus der Datenbank **325** und einer Vielzahl von Herunterladevorgängen über die langsame Kommunikationsverbindung führte.

In diesem Fall fragt jedoch die Benachrichtigungsmaschine **320** periodisch die Datenbank **325** nach einer Liste von Aktualisierungsereignissen für jedes der Datenbankelemente, die zu den magischen Zahlen in dem Wrapperplan **322** gehören, ab. Jedes Ereignis enthält Informationen, welche angeben, welches Objekt verändert wurde, und einige Informationen über die Veränderungen, wie etwa die Art der Veränderung (Umbenennung, Löschung, Änderung der Eigenschaften, Hinzufügen oder Löschen eines untergeordneten Elements etc.) und eventuell einige Hinweise, wie etwa den neuen und den alten Namen im Fall einer Umbenennungsoperation. Die Benachrichtigungsmaschine **320** verwendet den Wrapperplan **322**, um die magische Zahl des geänderten Objekts mit einem Wrapper-ID abzugleichen und fordert den zugehörigen Wrapper (in diesem Fall den Wrapper **372**) auf, das Aktualisierungsereignis zu verarbeiten. Diese Verarbeitung kann das erneute Laden des Lightweights **370** aus der Datenbank **325** einschließen, um den neuen oder geänderten Wert dieses Objekts zu erhalten. Ein derartiger erneuter Ladevorgang kann erforderlich sein, wenn das Lightweight **370** an der Durchführung der Veränderung an der Datenbank **325** nicht beteiligt war. Der Wrapper **372** kann das erneute Laden des Lightweights **370** verweigern, wenn beispielsweise eine erfaßte Veränderung eine alte oder nicht mehr aktuelle Veränderung ist, oder aus jedem anderen gewünschten Grund. Der Wrapper **372** benachrichtigt auch jede der Serverkomponenten **380** und **390** über die Veränderung, welche wiederum Mitteilungen, wie z. B. eine Windows-Mitteilung, an jeden der Clients **342** und **386** senden (welche diese Komponente abonniert haben), um die Veränderung anzuzeigen. Bei der Verarbeitung einer Ereignismitteilung von dem Wrapper **372** leiten die Serverkomponenten **380** und **390** das Event und eventuelle neue Daten an die Smart Proxys der jeweiligen Clients **342** und **386** weiter. Der Proxy **384** des ersten Clients **342** sendet eine Auffrischungsmitteilung an die Clientkomponente **382**, welche eine Auffrischung an einer Benutzerschnittstelle an dem ersten Client **344** verursachen kann. Wenn in diesem Fall die Benutzerschnittstelle aufgefrischt wird, gibt der Proxy **384** die neuen Daten an die Clientkomponente **382** auf und die Aktualisierung ist vollendet. Der zweite Client **386** kann die Mitteilung ignorieren, da der zweite Client **386** die Veränderung bereits hat.

Bei diesem Abonnementsystem mußte der erste Client **342** keine Datenbankanrufe durchführen und nur ein Anruf wurde (beispielsweise über eine langsame Kommunikationsverbindung) von dem Server **306** an den ersten Client **342** ausgeführt, um eine vollständige Erneuerung des geänderten Elements an den ersten Client **342** weiterzugeben. Selbstverständlich tritt dieser Prozeß für jedes Lightweight und für jeden Client, der ein Lightweight abonniert, auf, so daß jede Veränderung an der Datenbank **325** automatisch zu jedem Client zurückgesendet werden kann, der dieses Element abonniert, und zwar mit einem Minimalaufwand an Datenbanklesevorgängen und sehr wenigen Kommunikationsvorgängen über die langsame Kommunikationsverbindung.

Unter Bezug auf **Fig.** 14 wird die Operation des Verriegelungsmanagers **314** bei der Entscheidung von gleichzeitigem Zugriff auf den Cache **327** im Detail erläutert. Wie aus der vorstehenden Beschreibung deutlich wird, kann jedes der Lightweights in dem gemeinsam genutzten Cache **327** von einem oder mehreren Clientthreads gemeinsam genutzt werden. Als Resultat können viele Clients dasselbe Lightweight zur gleichen Zeit lesen. Typischerweise ist keine Verriegelung für diese Lesevorgänge erforderlich. Es ist jedoch möglich, in eine Situation zu geraten, in der zwei Clients gleichzeitig versuchen in ein Lightweight zu schreiben und somit in die Datenbank **325**, was Probleme, wie etwa eine gegenseitige Blockierung, verursachen kann. Um diese Probleme zu verhindern, verwaltet der Verriegelungsmanager **316** den Zugriff auf jedes der Lightweights, indem Verriegelungen ausgegeben werden und eine Verriegelungstabelle aufrechterhalten wird. Allgemein ausgedrückt arbeitet der Verriegelungsmanager **316** so, daß er Verriegelungsanfragen an Wrapper gewährt. Wenn ein Clientthread auf ein Lightweight zugreifen möchte, muß der Thread zunächst eine Lese- oder eine Schreibverriegelung an dem Wrapper erhalten. Wenn die Verriegelung nicht verfügbar ist, da beispielsweise ein anderer Clientthread im Zugriffsprozeß auf dieses Lightweight ist, hält der Verriegelungsmanager **316** den anfordernden Thread in Wartestellung, bis die Verriegelung frei wird.

In dem Beispiel von **Fig.** 14 versuchen der erste und der zweite Client **342** und **386** in die Beschreibung von Area_A zu schreiben. Hier versucht der zweite Client **386**, einen

Schreibvorgang an dem Lightweight **370** auszuführen, und in diesem Prozeß fordert er eine Verriegelung an dem Wrapper **372** von dem Verriegelungsmanager **316** an, die ihm gewährt wird. Wenn der erste Client **342** anschließend versucht, aus den Eigenschaften des Lightweights **370** zu lesen bzw. in diese zu schreiben, hält der Verriegelungsmanager **316** die Ausführung des ersten Clientthread **326** an, wenn dieser Thread versucht, eine Leseverriegelung oder eine Schreibverriegelung an dem Datenwrapper **372** zu erhalten. In der Zwischenzeit schreibt der zweite Clientthread **387** in das Lightweight **370** durch seinen eigenen Datenbankkontext **394**, um den Namen von Area_A in "Wilma" zu ändern. Wenn diese Schreiboperation vollendet ist, erlaubt der Verriegelungsmanager **316** dem ersten Clientthread **326**, fortzufahren und aus dem Lightweight **370** und somit aus der bzw. in die Datenbank **325** zu lesen oder in dieses zu schreiben. Es sei angemerkt, daß eine Schreibverriegelung verweigert oder zurückgehalten werden kann, bis eine Leseverriegelung gelöst ist. Es gibt jedoch gewöhnlich keinen Grund, eine Leseanforderung zurückzuhalten, wenn nur eine Leseverriegelung einem anderen Clientthread gewährt wurde. Ferner sei angemerkt, daß der Verriegelungsmanager **316** verwendet werden kann, um vorstehend beschriebene Reservierungsoperationen durchzuführen. Insbesondere kann eine Reservierungsverriegelung beispielsweise an dem Lightweight **370** von einem Client angefordert und diesem gewährt werden. Danach kann jeder Client lesend auf das Lightweight **370** zugreifen, jedoch nicht in das Lightweight schreiben. Wenn eine Reservierungsverriegelung gewährt wurde, kann der Verriegelungsmanager **316** veranlassen, daß der Wrapper **372** eine Mitteilung an den Client sendet, der eine Schreibverriegelung anfordert, daß keine Schreibvorgänge in diese Daten zulässig sind, da diese durch einen anderen Client reserviert wurden. Selbstverständlich kann der Verriegelungsmanager **316** auch ein Identifizierungskennzeichen des Client abgeben oder speichern, dem die Reservierungsverriegelung gewährt wurde, und diesen Client in die Lage versetzen, in das Lightweight **370** während eines Übertragungsvorganges zu schreiben.

Wenn mit Verriegelungen gearbeitet wird, besteht die Gefahr einer gegenseitigen Blockierung. Beispielsweise könnte ein erster Client eine Verriegelung an einer Komponente A erhalten und zur gleichen Zeit könnte ein zweiter Client eine Verriegelung an einer Komponente B erhalten. Anschließend könnte der erste Client versuchen, eine Verriegelung an der Komponente B zu erhalten, und würde aufgehalten, bis die Verriegelung an der Komponente B von dem zweiten Client gelöst wird. Wenn jedoch der zweite Client versucht, eine Verriegelung an der Komponente A zu erhalten, bevor die Verriegelung an der Komponente B gelöst wird, wird die Verriegelung an der Komponente A und Komponente B nie gelöst, was zu einer gegenseitigen Blokkierung führt. Um eine gegenseitige Blockierung in dem gemeinsam genutzten Cache **327** zu vermeiden, kann der Verriegelungsmanager **316** einer Einschränkung unterliegen, daß jeweils ein bestimmter Client eine Verriegelung erhalten bzw. auslösen kann und sie darauffolgend lösen muß, das heißt bevor eine weitere Verriegelung erhalten wird. In diesem Fall können jedoch zwei Clients Veränderungen überschreiben, die jeweils von dem anderen durchgeführt wurden, mit dem Resultat, daß der letzte Schreibvorgang bleibt. Beispielsweise modifiziert der erste Client Komponente A und der zweite Client modifiziert Komponente B. Der erste Client modifiziert anschließend Komponente B, und der zweite Client modifiziert Komponente A. Während eine gegenseitige Blockierung vermieden wurde, hat die Komponente A die Veränderungen des zweiten Clients und Komponente B hat die Veränderungen des ersten Clients.

Obgleich die Lightweights keine Hierarchie haben, können Verweise zwischen Lightweights vorhanden sein. Wenn dies der Fall ist, ist diese Tatsache eine mögliche Quelle einer gegenseitigen Blockierung, wenn beispielsweise eine Aktualisierungsoperation erfordert, daß beide Lightweights gleichzeitig verriegelt werden. Um dieses Problem zu vermeiden, kann der Verriegelungsmanager **316** eine Verriegelungsreihenfolge bei verwandten Lightweights festlegen. Wenn beispielsweise ein erstes Lightweight zu einem zweiten Lightweight gehört, wird das erste Lightweight zu der Verriegelungsliste des zweiten Lightweights hinzugefügt. Wenn ein Client das zweite Lightweight verriegeln möchte, muß der Client zunächst das erste Lightweight verriegeln. Der Verriegelungsmanager **316** kann eine Liste dieser Reihenfolge führen und die Reihenfolge bei Verriegelungsvorgängen durchsetzen. Um diese abhängige Verriegelung durchzusetzen, kann der Verriegelungsmanager nur eine Verriegelung pro Client zu einer bestimmten Zeit erlauben, falls die zugehörigen Objekte nicht einen abhängigen Baum bilden.

Wenn die Modifizierung einer Reihe von Komponenten in der Datenbank **325** in einer Datenbanktransaktion geschieht, ist es möglich, daß die Transaktion fehlschlägt und so eine Rückkehr auf den Status der Datenbank vor den Änderungen in der Datenbank **325** durchgeführt wurde, was dazu führt, daß die im Cache gespeicherten Komponenten (das heißt die Lightweights) mit den Komponenten, die in der Datenbank **325** gespeichert sind, nicht konsistent sind. Um einen Rückzugsmechanismus zu schaffen, kann ein Clientthread eine Liste von modifizierten Komponenten führen, zu welcher er die Komponenten hinzufügt, während die Datenbankmodifizierungstransaktion aktiv ist. Wenn eine Datenbanktransaktion mit mehreren Schritten abgebrochen wird oder das Commit fehlschlägt, wird die Liste durchgegangen und jede Komponente wird neu geladen, was bedeutet, daß die Eigenschaften einer Komponente neu geladen werden und eine Überprüfung auf neue/gelöschte untergeordnete Elemente erfolgt. Wenn eine derartige Liste aus modifizierten Komponenten in einer bekannten Ordnung (beispielsweise nach der Wrapper-ID geordnet) geführt wird, ist es möglich, eine Verriegelungstabelle zu erzeugen und mehrere Verriegelungen pro Operation des Clients auszuführen.

Um die Zeitdauer zu verringern, während der eine Lightweight verriegelt ist, kann ein Lightweight, wenn Eigenschaften in die Datenbank **325** zurückgeschrieben werden, zuerst das Komponentendatenobjekt klonen, das in dem Lightweight gespeichert ist. Der Klon wird mit den neuen Eigenschaftswerten versorgt und verwendet, um die Werte in der Datenbank **325** zu speichern. Sobald diese Operation vollendet ist, wird das ursprüngliche Komponentendatenobjekt in den Klon geändert oder zugunsten des Klons ersetzt. Die Verwendung einer derartigen Klonierungstechnik verringert die Zeit, während der der Cache **327** oder das Lightweight verriegelt ist, da das Lightweight entriegelt werden kann, nachdem der Klon erzeugt wurde, was die Gleichzeitigkeit des Zugriffs auf den Cache verbessert.

In dieser Situation kann jedoch eine Wettbewerbsbedingung vorliegen, in der zwei Clients versuchen, zur gleichen Zeit in dasselbe Lightweight zu schreiben. Insbesondere wird der Cache mit der Datenbank inkonsistent, wenn der erste Client seine Version der Komponenten in der Datenbank **325** speichert, bevor der zweite Client dies tut, aber den Cache (das heißt das Lightweight) nach dem zweiten Client abändert. Hier hat die Datenbank die Änderungen des zweiten Client, aber das Lightweight hat die Änderungen des ersten Client. Diese Situation kann entweder durch eine koordinierte Verriegelung vermieden werden, oder dadurch,

daß nur ein Klon für ein Lightweight zu einem gegebenen Zeitpunkt erlaubt wird. Alle diese Vorgänge können durch den Verriegelungsmanager **316** geleitet werden.

Wie vorstehend angeführt, kommen neue untergeordnete Lightweights entweder durch Laden dieser untergeordneten Elemente aus der Datenbank **325** ansprechend auf eine Anforderung eines Clients in Existenz, oder wenn ein Client ein neues untergeordnetes Element hinzufügt. Auf Wunsch kann zugelassen werden, daß Clients den Cache **327** soweit manipulieren, daß sie neue untergeordnete Komponentendatenobjekte schaffen/laden, bevor das übergeordnete Element verriegelt wird, so daß das übergeordnete Lightweight nur dann verriegelt werden muß, wenn ein untergeordnetes Element tatsächlich mit der Hierarchie verbunden wird oder dem übergeordneten Element zugeordnet wird. Mit anderen Worten können alle zu erstellenden Objekte, die mit untergeordneten Komponenten verbunden sind, in nicht gemeinsam genutzter Weise oder in einem lokalen Speicher erstellt werden, bis alle untergeordneten Elemente vollendet sind. Dann kann das übergeordnete Element verriegelt und modifiziert werden, so daß es die Zeiger zu diesen untergeordneten Elementen enthält, um dadurch diese untergeordneten Objekte in den gemeinsam genutzten Speicher zu verschieben. Als Resultat wird das übergeordnete Lightweight nur dann verriegelt, wenn es geändert wird, so daß es den Bezug zu neuen untergeordneten Komponenten hat, und nicht während der Erstellung dieser untergeordneten Objekte.

In **Fig.** 15 ist die Operation des Laufzeitbenachrichtigungsthreads dargestellt, der Clients über Veränderungen der Konfiguration eines Prozeßsteuersystems, die durch Laufzeitoperationen verursacht werden, benachrichtigt. In diesem Fall wird angenommen, daß ein Client **400** eine außer Dienst gestellte Einrichtungskomponente abonniert und daß die Clientkomponente **402**, das Proxy **404**, die Serverkomponente **406** der außer Dienst gestellten Einrichtung, der Wrapper **408** und das Lightweight **410** innerhalb des Clients **400** und des Servers **306** bereits eingerichtet sind. Ferner wird angenommen, daß der Wrapper **408** in dem Plan **340** registriert wurde, um Hinweise auf das Hinzufügen von neuen Steuereinrichtungen zu dem System zu erhalten, als der Wrapper **408** geschaffen wurde. In diesem Beispiel erfaßt die Benachrichtigungsmaschine **336** des Laufzeitbenachrichtigungsthreads das Hinzufügen einer neuen Steuereinrichtung (namens "CTLR001"), die über Laufzeitservices (wie z. B. eine automatisch erfassende Anwendung) hinzugefügt wurde, und sendet ein Aktualisierungsereignis an den Wrapper **408** der außer Dienst gestellten Einrichtungen, welches das Vorhandensein dieser neuen Steuereinrichtung angibt. Der Wrapper **408** für außer Dienst gestellte Einrichtungen antwortet auf das Aktualisierungsereignis, indem ein untergeordneter Wrapper **412** für die erfaßte Steuereinrichtung geschaffen wird, ein Lightweight **414** für die neue Steuereinrichtung geschaffen wird, und sofern möglich, die Steuereinrichtungsdaten als von den Laufzeitservices abgegeben in das Lightweight **414** gespeichert werden. Das Lightweight **414** kann diese Daten in der Datenbank **325** speichern, falls dies erwünscht ist, und das ID für den Datenwrapper **412** und die magische Zahl der Datenbankposition, an der die Information der neuen Steuereinrichtung gespeichert ist, in dem Wrapperplan **322** für den Benachrichtigungsthread registrieren. Der Wrapper **412** kann ferner eine Registrierung in dem Plan **340** für Aktualisierungen durchführen. Beim Laden der neuen Steuereinrichtung als ein neues untergeordnetes Element in dem gemeinsam genutzten Cache **327** wird ein Ereignis in der Serverkomponente für außer Dienst gestellte Einrichtungen **406** angezeigt, welche durch Erstellen einer neuen untergeordneten Steuereinrichtungsserverkomponente **416**, durch Einbetten eines Zeigers auf den neuen untergeordneten Datenwrapper **412** und durch Weitergeben der neuen Steuereinrichtungsdaten an den Client **400** reagiert. Der Client-Proxy **404** empfängt das neue untergeordnete Ereignis, speichert die neuen untergeordneten Informationen und setzt eine Auffrischungsmitteilung in die Benutzerschnittstelle an dem Client **400**. Wenn die Clientkomponente **402** für außer Dienst gestellte Einrichtungen die neuen untergeordneten Elemente lädt, baut die Clientkomponente **402** für außer Dienst gestellte Einrichtungen eine neue untergeordnete Steuereinrichtungskomponente **418** auf und schafft eine Verbindung zurück zu der Serverkomponente **416** für die Steuereinrichtungskomponente **418** unter Verwendung von Serverkomponentenidentifikationsinformationen, die von der Serverkomponente **406** für untergeordnete Einrichtungen zu dem Client **400** gesendet wurden. Ein Proxy **420** für die Steuereinrichtungskomponente **418** wird geschaffen und nimmt die Kommunikation mit der Serverkomponente **416** für die Steuereinrichtungskomponente **418** auf. Somit wird das Hinzufügen der Steuereinrichtung durch die Benachrichtigungsmaschine **336** erfaßt und diese Veränderung wird zu dem zugehörigen Datenwrapper **408** gesendet, der die in einem Lightweight gespeicherten Daten verändern kann oder das Erstellen oder das Löschen eines Lightweights und des zugehörigen Wrappers veranlassen kann. Das Erstellen eines neuen Wrappers und Lightweights führt zu der Erweiterung jedes Clientthreads, der das übergeordnete Objekt abonniert.

Abgesehen von den vorstehend beschriebenen Operationen können von dem Server **306** andere Aktivitäten oder Operationen ausgeführt werden. Beispielsweise kann ein Client eine Operation durchführen oder erstellen, bei welcher der Client eine Komponente erstellt, die in der Datenbank **325** zu speichern ist. Die mit dieser Erstellaktivität verbundenen Operationen sind sehr ähnlich dem Ladevorgang eines untergeordneten Elements, wobei der zusätzliche Schritt des Erstellens des zugrundeliegenden Datenbankobjekts in der Datenbank **325** anstelle des Ladens dieses Objekts aus der Datenbank **325** dazukommt. Hier erstellt der Client eine Komponente und sendet diese Komponente zu der Serverkomponente, die dem übergeordneten Element der erstellten Komponente zugehörig ist. Diese Tätigkeit veranlaßt den Wrappergenerator **312**, einen Wrapper und ein Lightweight als ein untergeordnetes Element des zugehörigen übergeordneten Wrappers und Lightweights zu schaffen, und das neue Lightweight wird mit der neuen Komponente geladen. Das Lightweight schreibt dann die neue Komponente in die Datenbank **325** und registriert die Datenbankposition und den Wrapper in dem Wrapperplan **322** in dem Benachrichtigungsthread. Eine neue Serverkomponente wird anschließend für den Wrapper erstellt und nimmt die Kommunikation mit den Client für diese neue Komponente auf.

Um eine Komponente aus der Datenbank **325** zu löschen, löscht ein Client zunächst die zugehörige Serverkomponente und gibt diese dann frei. Das Löschen der Serverkomponente setzt den zugehörigen Datenwrapper und das Lightweight in einen gelöschten Status, welcher anschließend die Komponente aus der Datenbank **325** löscht. Der Wrapper sendet eine Löschmitteilung an jede der anderen Bezug nehmenden Serverkomponenten (das heißt innerhalb anderer Clientthreads), welche wiederum die abonnierenden Clients über die Löschaktivität informieren. Im gelöschten Status wird der Wrapper aus der Konfigurationshierarchie entfernt und kann keine weiteren Datenbankoperationen durchführen. Der Wrapper kann jedoch verbleiben, bis alle Bezug nehmenden Serverkomponenten diesen freigegeben haben. Selbstverständlich geben die Bezug nehmenden Serverkomponenten diesen nicht frei, bis die zugehörigen Clients die

Freigabe erklärt haben, was auftritt, nachdem die Serverkomponenten die Clients über die Löschaktivität informiert haben. Selbstverständlich kann der Datenbankereignismechanismus schließlich eine Löschmitteilung erzeugen, welche durch die Benachrichtigungsmaschine des Benachrichtigungsthread zurück zu dem Wrapper geleitet wird. Da der Wrapper in einem gelöschten Zustand ist, ignoriert er diese Mitteilung.

Die Mutation bzw. Veränderung einer Konfigurationskomponente kann auftreten, da einige Operationen an einer Komponente verursachen, daß der Typ dieser Komponente geändert wird. Diese Mutation erfordert, daß die Komponente und der abhängige Baum von der Datenbank **325** neu geladen werden. Um diese Operation zu bewirken, wird die gesamte Parent/Child-Hierarchie, das heißt die Hierarchie von übergeordnetem Element und untergeordnetem Element für das Lightweight der mutierten Komponente aus der Datenbank **325** neu in die vorhandenen Lightweights geladen. Anschließend werden die Serverkomponenten neu erstellt und die neuen Werte innerhalb dieser Serverkomponenten werden anschließend mit den Clientkomponenten synchronisiert. Eine derartige Synchronisierung beinhaltet das Iterieren durch die untergeordneten Komponenten und Entfernen derjenigen, deren Typ sich verändert hat, und derjenigen, die nicht mehr vorhanden sind. Neue untergeordnete Elemente werden nach Bedarf geladen und hinzugefügt.

Selbstverständlich kann ein Client ohne weiteres eine Komponente innerhalb der Datenbank **325** freigeben oder das Abonnement lösen. In diesem Fall gibt der Client die zugehörige Serverkomponente frei, was verursacht, daß der zugehörige Wrapper einen Wrapperreferenzzählwert verringert. Wenn der Wrapperreferenzzählwert für einen Wrapper auf Null geht, nehmen keine Komponenten Bezug auf diesen Wrapper und somit abonnieren keine Clients diesen Wrapper. In diesem Fall wird der Wrapper entladen. Auf Wunsch kann der Wrapper eine Zeitauslösungsperiode abwarten, bevor er entladen wird, falls der Wrapper unmittelbar neu geladen werden muß. Beim Entladen zerstört der Wrapper sein zugehöriges Lightweight, löscht die Registrierung der Aktualisierungsbenachrichtigung in der Datenbank **325** und löscht die Registrierung aus dem Wrapperplan. Wenn ferner eine Komponente freigegeben wird, gibt sie ihr Lightweight und alle untergeordneten Lightweights/Wrapper frei und annulliert die rückverweisenden Zeiger von untergeordneten Lightweights/Wrappern.

Während die Operation des Servers **306** hier für zwei Clients, die auf eine oder zwei Komponenten zugreifen, beschrieben wurde, versteht sich von selbst, daß dieselbe Technik erweitert werden kann und für jede Anzahl von Clients und Clientthreads verwendet werden kann, die aus einer beliebigen Anzahl von Konfigurationsobjekten innerhalb einer Datenbank lesen und in diese schreiben. Während ferner die Technik zur Verwendung eines gemeinsam genutzten Cache zum Herstellen einer Abonnentenbeziehung für mehrere Clients hierin zur Verwendung beim Zugriff auf Konfigurationsdaten innerhalb eines Prozeßsteuersystems beschrieben wurde, versteht es sich, daß diese Technik in jedem anderen Datenbankzugriffssystem verwendet werden kann, einschließlich derjenigen, die nicht in Beziehung zu Konfigurationsdatenbanken stehen und derjenigen, die nicht in Beziehung zu Prozeßsteuersystemdatenbanken stehen.

Während ferner die hierin beschriebene Datenbankzugriffstechnik davon ausgeht, daß Clients mit einem Server auf einer Basis jeweils pro Komponente kommunizieren, kann das Marshaling von Konfigurationsbäumen verwendet werden, um eine verbesserte Kommunikation zwischen einem Client und einem Server zu schaffen. Insbesondere dann, wenn ein Client weiß, daß er einen gesamten Baum oder Baumabschnitt abrufen muß, ist es effizienter, einen Serveranruf durchzuführen, um den gesamten Baum zu erhalten, anstatt wiederholt eine Anzahl von untergeordneten Routinen auf jeder Ebene des Baumes abzurufen. Entgegengesetzt gilt ebenso, daß dann, wenn ein Client einen gesamten Baum zurück in den Server schreiben möchte, der gesamte Baum in einer Signalmitteilung gesendet werden kann. Selbstverständlich können diese Mitteilungen von dem Client und dem Server aufgefangen werden und die erforderlichen Objekte, um eine Abonnementbeziehung (wie vorstehend beschrieben) zu schaffen, können auf einmal erstellt werden, um dadurch alle Objekte innerhalb des Clients oder des Servers zu erstellen, die erforderlich sind, daß der Client den gesamten Baum oder Baumabschnitt abonnieren kann oder einen gesamten Baum oder Baumabschnitt schreiben kann.

Ferner versteht es sich, daß die Objekte und anderen Elemente und Schritte, die hierin beschrieben wurden, welche in dem Server und den Clients zu erstellen oder auszuführen sind, unter Verwendung jeder gewünschten Programmiertechnik oder -sprache implementiert werden können und daß diese Programme in beliebiger Weise innerhalb der hierin beschriebenen Clients, Server und Datenbanken in Speichern gespeichert und in Prozessoren ausgeführt werden können. Obgleich darüber hinaus die hierin beschriebene Datenzugriffstechnik vorzugsweise in Software implementiert wird, kann sie in Hardware, Firmware etc. implementiert werden und kann von jedem Prozessor implementiert werden, der dem Prozeßsteuersystem **10** zugehörig ist. So kann diese Technik in einer Standard-Mehrzweck-CPU oder auf einer speziell gestalteten Hardware oder Firmware nach Wunsch implementiert werden. Bei der Implementierung der Software können die Softwareroutinen in jeden computerlesbaren Speicher, wie z. B. auf einer Magnetplatte, einer Laserplatte, einer optischen Platte oder einem anderen Speichermedium, in einem RAM oder ROM eines Computers oder Prozessors etc. gespeichert werden. Entsprechend kann diese Software einem Benutzer oder einem Prozeßsteuersystem über jedes bekannte oder gewünschte Zulieferverfahren geliefert werden, darunter beispielsweise auf einer computerlesbaren Platte oder einem anderen transportablen Computerspeichermechanismus oder über einen Kommunikationskanal, wie etwa eine Telefonleitung, das Internet etc. (welche als gleich oder als austauschbar mit dem Weiterleiten derartiger Software über ein transportables Speichermedium betrachtet werden).

Patentansprüche

1. Konfigurationsdatenbanksystem zur Verwendung bei der Speicherung von Konfigurationsdaten, die zu einem Prozeßsteuersystem gehören, das eine Vielzahl von geographisch verteilten physischen Orten hat, welches Konfigurationsdatenbanksystem enthält:
eine an jedem der mehreren physischen Standorte befindliche Konfigurationsdatenbank, wobei jede der Konfigurationsdatenbanken so ausgelegt ist, daß sie ursprünglich einen unterschiedlichen Abschnitt der Konfigurationsdaten speichert;
ein Kommunikationsnetzwerk, das die Vielzahl der geographisch verteilten physischen Standorte miteinander in Kommunikationsverbindung setzt; und
eine Konfigurationsanwendung, die so ausgelegt ist, daß sie mit jeder der Datenbanken über das Kommunikationsnetzwerk kommuniziert und Daten von zwei oder mehr der Konfigurationsdatenbanken benutzt, um eine Konfigurationsaktivität durchzuführen.
2. Konfigurationsdatenbanksystem nach Anspruch 1,

dadurch gekennzeichnet, daß die Konfigurationsanwendung so ausgelegt ist, daß sie eine Teilmenge der Konfigurationsdaten innerhalb einer ersten der Konfigurationsdatenbanken abonniert und diese erste der Konfigurationsdatenbanken einen Datenbankserver enthält, der eine erste Routine hat, welche automatisch eine Veränderung an der Teilmenge der Konfigurationsdaten erkennt, die in der ersten Konfigurationsdatenbank gespeichert sind und welche die Konfigurationsanwendung abonniert, und eine zweite Routine, die automatisch die Konfigurationsanwendung über die Veränderung an der Teilmenge der Konfigurationsdaten benachrichtigt.

3. Konfigurationsdatenbanksystem nach Anspruch 2, dadurch gekennzeichnet, daß die zweite Routine automatisch die an der Teilmenge der Konfigurationsdaten, die in der ersten Konfigurationsdatenbank gespeichert sind, durchgeführte Veränderung an die Konfigurationsanwendung mitteilt, wenn die erste Routine die Veränderung an der Teilmenge der Konfigurationsdaten erfaßt.

4. Konfigurationsdatenbanksystem nach Anspruch 2, dadurch gekennzeichnet, daß die Konfigurationsanwendung einen Verriegelungsabschnitt enthält, der einen Verriegelungsbefehl zum Verriegeln eines Elements von Konfigurationsdaten innerhalb der ersten Konfigurationsdatenbank sendet, und dadurch, daß der Datenbankserver ferner eine Verriegelungsroutine enthält, welche das Element der Konfigurationsdaten innerhalb der ersten Konfigurationsdatenbank verriegelt, um zu verhindern, daß das Element der Konfigurationsdaten durch eine andere Konfigurationsanwendung als diejenige Konfigurationsanwendung, welche den Verriegelungsbefehl gesendet hat, verändert wird.

5. Konfigurationsdatenbanksystem nach Anspruch 2, ferner enthaltend eine zweite Konfigurationsanwendung, wobei der Datenbankserver eine zweite Routine enthält, welche gleichzeitigen Zugriff auf ein bestimmtes Element von Konfigurationsdaten innerhalb der ersten Konfigurationsdatenbank für die erste und die zweite Konfigurationsanwendung gewährt, und wobei der Datenbankserver ferner eine Verriegelungsroutine enthält, welche das bestimmte Element der Konfigurationsdaten innerhalb der ersten Konfigurationsdatenbank verriegelt, wenn die erste Konfigurationsanwendung in das bestimmte Element von Konfigurationsdaten schreibt, um dadurch zu verhindern, daß die zweite Konfigurationsanwendung das bestimmte Element von Konfigurationsdaten verändert, wenn die erste Konfigurationsanwendung in das bestimmte Element von Konfigurationsdaten schreibt.

6. Konfigurationsdatenbanksystem nach Anspruch 1, dadurch gekennzeichnet, daß die Konfigurationsdatenbanken in einer Hierarchie eingerichtet sind, die mindestens zwei Konfigurationsdatenbanken in einer niedrigeren Ebene und mindestens eine Konfigurationsdatenbank in einer höheren Ebene hat, und wobei jede der Konfigurationsdatenbanken in der niedrigeren Ebene mit der Konfigurationsdatenbank in der höheren Ebene in Kommunikationsverbindung steht.

7. Konfigurationsdatenbanksystem nach Anspruch 6, dadurch gekennzeichnet, daß die untere Ebene Zonen enthält, die zu verschiedenen physischen Orten gehören, wobei die Konfigurationsdaten, die zu einer ersten Zone gehören, in der Konfigurationsdatenbank in der ersten Zone gespeichert werden, und die Konfigurationsdaten, die zu der ersten und der zweiten Zone gehören, in der Konfigurationsdatenbank in der höheren Ebene in der Hierarchie gespeichert werden, die mit der ersten und der zweiten Zone in Kommunikationsverbindung steht.

8. Konfigurationsdatenbanksystem nach Anspruch 6, ferner enthaltend Software, welche die Verwendung desselben Namens in einer der Konfigurationsdatenbanken innerhalb der unteren Ebene und der Konfigurationsdatenbank in der höheren Ebene erfaßt.

9. Konfigurationsdatenbanksystem nach Anspruch 1, dadurch gekennzeichnet, daß die Konfigurationsanwendung eine Suchroutine enthält, die so ausgelegt ist, daß sie eine erste der Konfigurationsdatenbanken durchsucht und eine zweite der Konfigurationsdatenbanken durchsucht.

10. Konfigurationsdatenbanksystem nach Anspruch 1, dadurch gekennzeichnet, daß eine der Konfigurationsdatenbanken eine lokale Kopie eines Elements speichert, das ursprünglich in einer anderen Konfigurationsdatenbank gespeichert ist.

11. Konfigurationsdatenbanksystem nach Anspruch 1, dadurch gekennzeichnet, daß das Kommunikationsnetzwerk eine langsame Kommunikationsverbindung zwischen zwei der Konfigurationsdatenbanken enthält.

12. Verteiltes Konfigurationsdatenbanksystem, das so ausgelegt ist, daß es in einem Prozeßsteuersystem mit mehreren physischen Orten verwendet wird, enthaltend:

eine erste Konfigurationsdatenbank, die an einem ersten physischen Ort angeordnet ist, welche einen ersten Abschnitt von Konfigurationsdaten für das Prozeßsteuersystem speichert;

eine zweite Konfigurationsdatenbank, die an einem zweiten physischen Ort angeordnet ist, welche einen zweiten Abschnitt der Konfigurationsdaten für das Prozeßsteuersystem speichert, wobei der erste Abschnitt der Konfigurationsdaten von dem zweiten Abschnitt der Konfigurationsdaten verschieden ist; und

ein Kommunikationsnetzwerk, das den ersten physischen Ort und den zweiten physischen Ort unter Verwendung einer langsamen Kommunikationsverbindung verbindet,

wobei die erste und die zweite Konfigurationsdatenbank so ausgelegt sind, daß sie mit Benutzern an dem ersten und dem zweiten physischen Ort kommunizieren.

13. Verteiltes Konfigurationsdatenbanksystem nach Anspruch 12, dadurch gekennzeichnet, daß die zweite Konfigurationsdatenbank so ausgelegt ist, daß sie Konfigurationsdaten in der ersten Konfigurationsdatenbank abonniert, wobei die erste Konfigurationsdatenbank einen Datenbankserver enthält, der eine erste Routine hat, die automatisch eine Veränderung an Konfigurationsdaten erfaßt, die in der ersten Konfigurationsdatenbank gespeichert sind, welche die zweite Konfigurationsdatenbank abonniert, und eine zweite Routine, die automatisch die zweite Konfigurationsdatenbank über die Veränderung an den Konfigurationsdaten, die in der ersten Konfigurationsdatenbank gespeichert sind, welche die zweite Konfigurationsdatenbank abonniert, benachrichtigt.

14. Verteiltes Konfigurationsdatenbanksystem nach Anspruch 13, bei welchem die zweite Routine automatisch die an den Konfigurationsdaten, die in der ersten Konfigurationsdatenbank gespeichert sind, vorgenommenen Veränderungen an die zweite Konfigurationsdatenbank mitteilt, wenn die erste Routine die Veränderung an den in der ersten Konfigurationsdatenbank gespeicherten Konfigurationsdaten, welche die zweite

Konfigurationsdatenbank abonniert, erfaßt.

15. Verteiltes Konfigurationsdatenbanksystem nach Anspruch 12, ferner enthaltend eine erste und eine zweite Konfigurationsanwendung und einen Datenbankserver, der der ersten Konfigurationsdatenbank zugeordnet ist, wobei der Datenbankserver einen gemeinsam genutzten Cache enthält, der gleichzeitigen Zugriff auf jedes bestimmte Element von Konfigurationsdaten in der ersten Konfigurationsdatenbank für die erste und die zweite Konfigurationsanwendung gewährt, und eine Verriegelungsroutine enthält, welche das bestimmte Element der Konfigurationsdaten in der ersten Konfigurationsdatenbank verriegelt, wenn die erste Konfigurationsanwendung in das bestimmte Element von Konfigurationsdaten schreibt, um dadurch zu verhindern, daß die zweite Konfigurationsanwendung das bestimmte Element von Konfigurationsdaten verändert, wenn die erste Konfigurationsanwendung in das bestimmte Element von Konfigurationsdaten schreibt.

16. Verteiltes Konfigurationsdatenbanksystem nach Anspruch 12, ferner enthaltend eine dritte Konfigurationsdatenbank, die an einem dritten physischen Ort befindlich ist, wobei die erste, die zweite und die dritte Konfigurationsdatenbank in einer Hierarchie eingerichtet sind, in der die erste und die zweite Konfigurationsdatenbank in einer niederen Ebene sind und die dritte Konfigurationsdatenbank in einer höheren Ebene ist, und wobei jede der ersten und zweiten Konfigurationsdatenbanken in der niederen Ebene mit der dritten Konfigurationsdatenbank in der höheren Ebene über das Kommunikationsnetzwerk in Kommunikationsverbindung steht.

17. Verteiltes Konfigurationsdatenbanksystem nach Anspruch 16, dadurch gekennzeichnet, daß die untere Ebene eine erste und eine zweite Zone enthält, wobei Konfigurationsdaten, die zu einer ersten Zone gehören, in der ersten Konfigurationsdatenbank gespeichert sind, Konfigurationsdaten, die zu der zweiten Zone gehören, in der zweiten Konfigurationsdatenbank gespeichert sind, und Konfigurationsdaten, die sowohl zu der ersten als auch zu der zweiten Zone gehören, in der dritten Konfigurationsdatenbank gespeichert sind.

18. Verteiltes Konfigurationsdatenbanksystem nach Anspruch 16, ferner enthaltend eine Routine, die die Verwendung desselben Namens innerhalb der ersten und der dritten Konfigurationsdatenbank oder innerhalb der zweiten und der dritten Konfigurationsdatenbank erfaßt.

19. Verfahren zum Speichern und Verwenden von Konfigurationsdaten, die sich auf ein Prozeßsteuersystem beziehen, wenn das Prozeßsteuersystem zwei oder mehr physische Orte hat, die geographisch getrennt sind, welches Verfahren die Schritte enthält:

Speichern eines unterschiedlichen Abschnittes der Konfigurationsdaten in jeweils einer Vielzahl von Konfigurationsdatenbanken, wobei zwei der Konfigurationsdatenbanken an verschiedenen physischen Orten befindlich sind;

Vorsehen einer Kommunikationsverbindung zwischen jeder der Konfigurationsdatenbanken; und

Zugreifen auf verschiedene Kommunikationsdaten von zwei oder mehr der Konfigurationsdatenbanken zur gleichen Zeit, um eine Konfigurationsaktivität auszuführen.

20. Verfahren zur Speicherung und Nutzung von Konfigurationsdaten nach Anspruch 19, dadurch gekennzeichnet, daß der Schritt des Zugreifens auf verschiedene Konfigurationsdaten den Schritt des Verwendens einer Anwendung enthält, um die Konfigurationsdaten, auf die Zugriff besteht, von den zwei oder mehr der Konfigurationsdatenbanken zu abonnieren und automatisch alle Veränderungen, die an den im Zugriff befindlichen Konfigurationsdaten durchgeführt werden, von den zwei oder mehr Konfigurationsdatenbanken an die Anwendung zu senden.

21. Verfahren zur Speicherung und Nutzung von Konfigurationsdaten nach Anspruch 19, ferner enthaltend den Schritt des Einrichtens einer Hierarchie unter den Konfigurationsdatenbanken, wobei die Hierarchie zwei Konfigurationsdatenbanken in einer unteren Ebene und eine Konfigurationsdatenbank in einer höheren Ebene enthält, und des Einrichtens einer Kommunikationsverbindung zwischen jeder der zwei Konfigurationsdatenbanken in der unteren Ebene und der einen Konfigurationsdatenbank in der höheren Ebene.

22. Verfahren zur Speicherung und Nutzung von Konfigurationsdaten nach Anspruch 21, ferner enthaltend der Verwendung von einzigartigen Namen von Konfigurationsdaten in der einen Konfigurationsdatenbank in der höheren Ebene der Hierarchie und einer der beiden Konfigurationsdatenbanken in der unteren Ebene der Hierarchie, wobei derselbe Name in den beiden Konfigurationsdatenbanken in der unteren Ebene der Hierarchie verwendet werden kann.

23. Datenbankserver, der so ausgelegt ist, daß er mehreren Clients gleichzeitigen Zugriff auf eine Datenbankkomponente bietet, die in einer Datenbank gespeichert ist, welcher Datenbankserver enthält:

einen gemeinsam genutzten Cache, der einen Speicher enthält, der mit der Datenbank in Kommunikationsverbindung steht, wobei der Speicher eine Kopie der Datenbankkomponente enthält;

einen Speicherzugang, der zu dem Speicher gehört, wobei der Speicherzugang den Zugriff auf den Speicher steuert; und

ein Clientthread für jeden der mehreren Clients, wobei jeder der Clientthreads eine Serverkomponente enthält, die mit dem Speicherzugang kommuniziert, um Zugriff auf den Speicher zu erhalten, und die mit einem der mehreren Clients mit Bezug auf die Datenbankkomponente, die in dem Speicher gespeichert ist, kommuniziert;

wobei der Speicherzugang mit einer einzelnen Serverkomponente kommuniziert, wenn nur einer der mehreren Clients auf die Datenbankkomponente zugreift, und der Speicherzugang mit zwei oder mehr Serverkomponenten kommuniziert, wenn zwei oder mehr Clients auf die Datenbankkomponente zugreifen.

24. Datenbankserver nach Anspruch 23, ferner enthaltend einen Benachrichtigungsthread, der eine Benachrichtigungsmaschine hat, die eine Veränderung erfaßt, die an der Datenbankkomponente in der Datenbank durchgeführt wurde und die den Speicherzugang über die erfaßte Veränderung benachrichtigt.

25. Datenbankserver nach Anspruch 24, dadurch gekennzeichnet, daß der Speicherzugang die Serverkomponenten, die dem Speicherzugang die erfaßte Veränderung mitteilen, benachrichtigt, wobei die benachrichtigten Serverkomponenten die Clients benachrichtigen, mit welchen die benachrichtigten Serverkomponenten in Kommunikation über die erfaßte Veränderung stehen.

26. Datenbankserver nach Anspruch 24, dadurch gekennzeichnet, daß der Benachrichtigungsthread ferner einen Plan enthält, der die Datenbankkomponente, die

in der Datenbank gespeichert ist, in Bezug auf den Speicherzugang verzeichnet, wobei die Benachrichtigungsmaschine den Plan verwendet, um den Speicherzugang über die erfaßte Veränderung zu benachrichtigen.

27. Datenbankserver nach Anspruch 26, dadurch gekennzeichnet, daß der Speicherzugang ein einzigartiges Identifizierungskennzeichen enthält und das einzigartige Identifizierungskennzeichen in dem Plan einträgt, wobei der Speicher ein Transaktionsobjekt verwendet, um die Datenbankkomponente aus der Datenbank auszulesen oder die Datenbankkomponente in die Datenbank zu schreiben, und wobei das Transaktionsobjekt einen Datenbankkomponentenort für die Datenbankkomponente in dem Plan registriert, der mit dem einzigartigen Kommunikationskennzeichen zu verbinden ist.

28. Datenbankserver nach Anspruch 24, ferner enthaltend einen Laufzeitthread, der eine Laufzeitbenachrichtigungsmaschine enthält, welche Veränderungen erfaßt, die bezüglich der Datenbankkomponente durchgeführt wurden, welche von außerhalb der Datenbank ausgeführten Anwendungen durchgeführt wurden, und welcher den Speicherzugang über die erfaßte Veränderung an der Datenbankkomponente benachrichtigt.

29. Datenbankserver nach Anspruch 23, dadurch gekennzeichnet, daß die Serverkomponente jedes der Clientthreads in der Lage ist, aus dem Speicher zu lesen und in den Speicher zu schreiben.

30. Datenbankserver nach Anspruch 23, dadurch gekennzeichnet, daß eine der Serverkomponenten mit einem der mehreren Clients über eine langsame Kommunikationsverbindung in Kommunikation steht.

31. Datenbankserver nach Anspruch 23, dadurch gekennzeichnet, daß die langsame Kommunikationsverbindung eine Satellitenkommunikationsverbindung ist.

32. Datenbankserver nach Anspruch 23, ferner enthaltend einen Verriegelungsmanager, der veranlaßt, daß der Speicherzugang Schreibvorgänge in dem Speicher verhindert.

33. Datenbankserver nach Anspruch 32, dadurch gekennzeichnet, daß der Verriegelungsmanager den Speicherzugang veranlaßt, Schreibvorgänge in dem Speicher von jedem der Clients mit Ausnahme eines der Clients zu verhindern, wenn dieser eine der Clients in den Speicher schreibt.

34. Datenbankserver nach Anspruch 23, dadurch gekennzeichnet, daß die Datenbank eine Konfigurationsdatenbank ist und die Datenbankkomponente eine Konfigurationskomponente ist, die in Beziehung zu der Konfiguration eines Prozeßsteuernetzwerkes steht.

35. Datenbankserver, der dazu ausgelegt ist, mehreren Clients gleichzeitigen Zugriff auf eine Vielzahl von Datenbankkomponenten zu gewähren, die in einer Datenbank gespeichert sind, welcher Datenbankserver enthält:

einen gemeinsam genutzten Cache, der eine Vielzahl von Speichern enthält, die mit der Datenbank in Kommunikationsverbindung stehen, wobei jeder der Speicher eine Kopie einer anderen Datenbankkomponente enthält;

einen Speicherzugang, der mit jedem der Speicher verbunden ist, wobei jeder der Speicherzugänge den Zugriff auf den zugehörigen Speicher steuert; und

einen Clientthread für jeden der mehreren Clients, wobei jeder der Clientthreads eine Serverkomponente enthält, die mit einem der Speicherzugänge kommuniziert, um Zugriff auf den zu diesem einen Speicherzugang gehörigen Speicher zu erhalten, und die mit einem der mehreren Clients im Hinblick auf die in dem Speicher, der dem einen der Speicherzugänge zugehörig ist, gespeicherte Datenbankkomponente kommuniziert;

wobei jeder der Speicherzugänge mit einer einzelnen Serverkomponente in Kommunikation steht, wenn nur einer der mehreren Clients auf die in dem Speicher, der zu dem Speicherzugang gehört, gespeicherte Datenbankkomponente zugreift, und wobei jeder der Speicherzugänge mit zwei oder mehr Serverkomponenten kommuniziert, wenn zwei oder mehr der mehreren Clients auf die in dem zu dem Speicherzugang gehörigen Speicher gespeicherte Datenbankkomponente zugreifen.

36. Datenbankserver nach Anspruch 35, ferner enthaltend einen Benachrichtigungsthread, der eine Benachrichtigungsmaschine hat, die eine Veränderung erfaßt, die an einer der Datenbankkomponenten vorgenommen wurde, und die einen der Speicherzugänge über die erfaßte Veränderung benachrichtigt.

37. Datenbankserver nach Anspruch 36, dadurch gekennzeichnet, daß der eine der Speicherzugänge die Serverkomponenten, die mit dem einen der Speicherzugänge in Kommunikation stehen, über die erfaßte Veränderung benachrichtigt, wobei die benachrichtigten Serverkomponenten die Clients, mit welchen die benachrichtigten Serverkomponenten kommunizieren, über die erfaßte Veränderung benachrichtigen.

38. Datenbankserver nach Anspruch 37, dadurch gekennzeichnet, daß die benachrichtigten Serverkomponenten die erfaßte Veränderung an Clients weitergeben, mit welchen die benachrichtigten Serverkomponenten kommunizieren.

39. Datenbankserver nach Anspruch 35, ferner enthaltend einen Verriegelungsmanager, der veranlaßt, daß jeder der Speicherzugänge Schreibvorgänge in dem zugängigen Speicher verhindert.

40. Datenbankserver nach Anspruch 39, dadurch gekennzeichnet, daß die beiden Speicherzugänge eine Parent/Child-Beziehung, also eines übergeordneten Elements und eines untergeordneten Elements haben, so daß einer der beiden Speicherzugänge ein übergeordneter Speicherzugang ist und der andere der beiden Speicherzugänge ein untergeordneter Speicherzugang ist, wobei der Verriegelungsmanager eine Verriegelung des untergeordneten Speicherzugangs veranlaßt, wenn der Verriegelungsmanager eine Verriegelung des übergeordneten Speicherzugangs veranlaßt.

41. Datenbankserver, der zur Verwendung in einem Prozeßsteuersystem ausgelegt ist, um mehreren Clients gleichzeitigen Zugriff auf Datenbankkomponenten zu gewähren, die in einer Datenbank gespeichert sind, welcher Datenbankserver enthält:

einen gemeinsam genutzten Cache;

ein erstes Element, das in dem gemeinsam genutzten Cache einen Speicher für die Datenbankkomponente schafft, wenn auf die Datenbankkomponente von einem oder mehreren der mehreren Clients zugegriffen wird; und

eine Clientthread-Routine, die einen Clientthread für jeden der mehreren Clients erstellt, wobei jeder der Clientthreads eine Serverkomponente enthält, die mit einem der Clients kommuniziert und die mit dem Speicher kommuniziert, um dadurch Zugriff durch einen der Clients auf die in dem Speicher gespeicherte Datenbankkomponente zu ermöglichen;

wobei der Speicher so ausgelegt ist, daß er mit einer einzelnen Serverkomponente kommuniziert, wenn nur

einer der mehreren Clients auf die Datenbankkomponente zugreift, und wobei der Speicher so ausgelegt ist, daß er mit zwei oder mehr Serverkomponenten kommuniziert, wenn zwei oder mehr der mehreren Clients auf die Datenbankkomponente zugreifen.

42. Datenbankserver nach Anspruch 41, dadurch gekennzeichnet, daß das erste Element ferner einen Speicherzugang schafft, der zu dem Speicher gehört, wobei der Speicherzugang so ausgelegt ist, daß er mit den Serverkomponenten in zwei oder mehr der Clientthreads kommuniziert, um Zugriff auf den zugehörigen Speicher durch jede der Serverkomponenten in den zwei oder mehr Clientthreads zu gewähren, wenn zwei oder mehr Clients auf die Datenbankkomponente zugreifen.

43. Datenbankserver nach Anspruch 42, dadurch gekennzeichnet, daß der Speicher ein transaktionales Objekt nutzt, um auf die Datenbankkomponente in der Datenbank zuzugreifen.

44. Datenbankserver nach Anspruch 42, ferner enthaltend einen Benachrichtigungsthread, der eine Benachrichtigungsmaschine und einen Plan hat, wobei die Benachrichtigungsmaschine eine Veränderung an der Datenbankkomponente innerhalb der Datenbank erfaßt und den Plan benutzt, den Speichereingang über die erfaßte Veränderung zu benachrichtigen.

45. Datenbankserver nach Anspruch 44, dadurch gekennzeichnet, daß der Speichereingang alle Serverkomponenten, mit welchen der Speichereingang in Verbindung steht, über die erfaßte Veränderung an der Datenbankkomponente in Kenntnis setzt, wobei die benachrichtigten Serverkomponenten die Clients, mit welchen die benachrichtigten Serverkomponenten kommunizieren, über die erfaßte Veränderung benachrichtigen.

46. Datenbankserver nach Anspruch 44, dadurch gekennzeichnet, daß der Speicherzugang den Speicher veranlaßt, die Datenbankkomponente aus der Datenbank auszulesen, wenn die Benachrichtigungsmaschine den Speicherzugang über die erfaßte Veränderung an der Datenbankkomponente innerhalb der Datenbank benachrichtigt.

47. Datenbankserver nach Anspruch 44, dadurch gekennzeichnet, daß der Benachrichtigungsthread eine Veränderungsliste benutzt, die von der Datenbank erzeugt wird, um die Änderung an der Datenbankkomponente innerhalb der Datenbank zu erfassen.

48. Datenbankserver nach Anspruch 44, ferner enthaltend einen Laufzeitservicebenachrichtigungsthread, der eine Veränderung an der Datenbankkomponente, basierend auf der Operation von Laufzeitanwendungen erfaßt, die in dem Prozeßsteuersystem ausgeführt werden, und der den Speicherzugang über die erfaßte Veränderung benachrichtigt.

49. Datenbankserver nach Anspruch 41, dadurch gekennzeichnet, daß mindestens eine der Serverkomponenten so ausgelegt ist, daß sie mit einem der mehreren Clients über eine Satellitenkommunikationsverbindung kommuniziert.

50. Datenbankserver nach Anspruch 41, dadurch gekennzeichnet, daß mindestens eine der Serverkomponenten so ausgelegt ist, daß sie mit einem der mehreren Clients über eine zelluläre Kommunikationsverbindung kommuniziert.

51. Datenbankserver nach Anspruch 41, dadurch gekennzeichnet, daß mindestens eine der Serverkomponenten so ausgelegt ist, daß sie mit einem der mehreren Clients über eine drahtlose Kommunikationsverbindung kommuniziert.

52. Datenbankserver nach Anspruch 41, dadurch gekennzeichnet, daß mindestens eine der Serverkomponenten so ausgelegt ist, daß sie mit einem der mehreren Clients über eine Telefonkommunikationsverbindung kommuniziert.

53. Datenbankserver nach Anspruch 41, ferner enthaltend einen Verriegelungsmanager, der den Speicher verriegelt, wenn einer der mehreren Clients auf den Speicher zugreift, um den Zugriff auf den Speicher durch andere der mehreren Clients zu verhindern.

54. Datenbankserver nach Anspruch 41, ferner enthaltend einen Kontextspeicher, wobei der Speicher einen ersten Abschnitt des Kontextspeichers nutzt, wenn er mit der Datenbank im Namen eines ersten der Clients kommuniziert, und einen zweiten, davon verschiedenen Abschnitt des Kontextspeichers nutzt, wenn er mit der Konfigurationsdatenbank namens eines zweiten Clients kommuniziert.

55. Verfahren um mehreren Clients Zugriff auf eine Datenbankkomponente zu gewähren, die in einer Datenbank gespeichert ist, welches Verfahren die Schritte enthält:

Erstellen eines Speicherobjektes in einem gemeinsam genutzten Cache, wobei das Speicherobjekt mit der Datenbank kommuniziert, um im Hinblick auf die Datenbankkomponente aus der Datenbank zu lesen oder in die Datenbank zu schreiben;

Verwenden eines ersten Clientthreads, um eine Kommunikation zwischen einem ersten der Clients und der Datenbankkomponente zu schaffen, wobei der Schritt des Verwendens des ersten Clientthreads die Schritte enthält;

Erstellen einer ersten Kommunikationskomponente, die mit dem ersten Client hinsichtlich der Datenbankkomponente kommuniziert; und

Schaffen einer Verbindung zwischen der ersten Kommunikationskomponente und dem Speicherobjekt; und

Verwenden eines zweiten Clientthreads, um die Kommunikation zwischen einem der zweiten der Clients und der Datenbankkomponente herzustellen, wobei der Schritt des Verwendens des zweiten Clientthreads die Schritte enthält;

Erstellen einer zweiten Kommunikationskomponente, die mit dem zweiten Clients hinsichtlich der Datenbankkomponente kommuniziert; und

Erstellen einer Verbindung zwischen der zweiten Kommunikationskomponente und dem Speicherobjekt.

56. Verfahren nach Anspruch 55, dadurch gekennzeichnet, daß der Schritt des Erstellens des Speicherobjekts den Schritt des Erstellens eines Speicherobjektzugangs enthält, der zu dem Speicherobjekt gehört, wobei der Speicherobjektzugang so ausgelegt ist, daß er mit den Serverkomponenten in dem ersten und dem zweiten Clientthread kommuniziert, um Zugriff auf das Speicherobjekt für jede der Speicherkomponenten in dem ersten und dem zweiten Clientthread zu gewähren.

57. Verfahren nach Anspruch 56, enthaltend den Schritt der Verwendung eines Transaktionsobjektes für den Zugriff auf die Datenbankkomponente innerhalb der Datenbank und um Kommunikation zwischen dem Speicherobjekt und der Datenbank zu schaffen.

58. Verfahren nach Anspruch 56, ferner enthaltend den Schritt des Erfassens einer Veränderung an der Datenbankkomponente innerhalb der Datenbank und des Benachrichtigens des Speicherobjektzugangs über die Veränderung.

59. Verfahren nach Anspruch 58, ferner enthaltend den

Schritt des Verwendens des Speicherobjektzugangs zur Benachrichtigung aller Serverkomponenten, mit welchem der Speicherobjektzugang kommuniziert, über die erfaßte Veränderung an der Datenbankkomponente, und enthaltend den Schritt des Mitteilens der erfaßten Veränderung an der Datenbankkomponente von den benachrichtigten Serverkomponenten an die Clients, mit welchen die benachrichtigten Serverkomponenten kommunizieren.

60. Verfahren nach Anspruch 59, ferner enthaltend den Schritt des Veranlassens des Speicherobjekts, die Datenbankkomponente aus der Datenbank auszulesen, wenn die Veränderung an der Datenbankkomponente erfaßt wird.

61. Verfahren nach Anspruch 59, dadurch gekennzeichnet, daß der Schritt des Bekanntgebens der erfaßten Veränderung von den benachrichtigten Serverkomponenten an die Clients den Schritt des Verwendens einer Satellitenkommunikationsverbindung einschließt, um die Kommunikation zwischen einer der Serverkomponenten und einem der Clients herzustellen.

62. Verfahren nach Anspruch 59, dadurch gekennzeichnet, daß der Schritt des Bekanntgebens der erfaßten Veränderung von den benachrichtigten Serverkomponenten an die Clients den Schritt des Verwendens einer drahtlosen Kommunikationsverbindung einschließt, um die Kommunikation zwischen einer der Serverkomponenten und einem der Clients herzustellen.

63. Verfahren nach Anspruch 59, dadurch gekennzeichnet, daß der Schritt des Bekanntgebens der erfaßten Veränderung von den benachrichtigten Serverkomponenten an die Clients den Schritt des Verwendens einer Telefonleitungs-Kommunikationsverbindung einschließt, um die Kommunikation zwischen einer der Serverkomponenten und einem der Clients herzustellen.

64. Verfahren nach Anspruch 59, ferner enthaltend den Schritt des Verriegelns des Speicherobjekts, wenn einer der mehreren Clients auf das Speicherobjekt zugreift, um den Zugriff auf das Speicherobjekt durch andere der mehreren Clients zu verhindern.

65. Verfahren zur Verwendung eines Datenbankservers, um einer Vielzahl von Clients gleichzeitigen Zugriff auf Komponenten zu ermöglichen, die in einer Datenbank gespeichert sind, welche einen gemeinsam genutzten Cache hat, welches Verfahren die Schritte enthält:

Durchführen der folgenden drei Schritte, wenn ein Client Zugriff auf eine Komponente innerhalb der Datenbank anfordert;

(1) Schaffen eines Kommunikationskomponentenobjekts innerhalb eines Datenbankservers, das mit dem Client hinsichtlich der Komponente kommuniziert;

(2) Feststellen, ob ein gemeinsam genutztes Speicherobjekt innerhalb des gemeinsam genutzten Cache für die Komponente eingerichtet wurde, und Erstellen des gemeinsam genutzten Speicherobjekts für die Komponente innerhalb des gemeinsam genutzten Cache, wenn das gemeinsam genutzte Speicherobjekt nicht in dem gemeinsam genutzten Cache eingerichtet wurde; und

(3) Vorsehen einer Kommunikationsverbindung zwischen dem gemeinsam genutzten Speicherobjekt und der Kommunikationskomponente; und

Verwenden des gemeinsam genutzten Speicherobjekts, um aus der Datenbank zu lesen und in diese zu schreiben, um dadurch die Komponente aus der Datenbank auszulesen und in die Komponente in der Datenbank zu schreiben;

wobei dann, wenn zwei oder mehr Clients auf dieselbe Komponente zugreifen, zwei oder mehr Kommunikationskomponentenobjekte mit demselben gemeinsam genutzten Speicherobjekt kommunikativ verbunden werden.

66. Verfahren nach Anspruch 65, ferner enthaltend den Schritt des Erfassens einer Veränderung an einer der Komponenten und Benachrichtigens jedes der Clients, die auf eine der Komponenten zugreifen, über die erfaßte Veränderung.

67. Verfahren nach Anspruch 66, ferner enthaltend einen Schritt, in dem veranlaßt wird, daß das gemeinsam genutzte Speicherobjekt für die eine der Komponenten die veränderte Komponente aus der Datenbank ausliest.

68. Verfahren nach Anspruch 66, ferner enthaltend den Schritt des Verriegelns eines der gemeinsam genutzten Speicherobjekte, wenn eine der Kommunikationskomponenten auf dieses eine der gemeinsam genutzten Speicherobjekte zugreift, um dadurch zu verhindern, daß andere Kommunikationsobjekte auf die gemeinsam genutzte Speicherkomponente zugreifen, wenn dieses eine der Kommunikationsobjekte Zugriff auf das gemeinsam genutzte Speicherobjekt hat.

69. Prozeßsteuersystem, enthaltend:

eine Datenbank, die an einem ersten physischen Ort angeordnet ist, wobei die Datenbank Datenbankkomponenten speichert;

eine Vielzahl von Client-Anwendungen, wobei eine der Client-Anwendungen an einem zweiten physischen Ort angeordnet ist, der im wesentlichen von dem ersten physischen Ort geographisch getrennt ist;

eine Kommunikationsverbindung zwischen dem ersten physischen Ort und dem zweiten physischen Ort; und

einen Datenbankserver, der für die Vielzahl der Client-Anwendungen Zugriff auf die Datenbankkomponenten in der Datenbank bietet, welcher Datenbankserver enthält;

einen gemeinsam genutzten Cache, der eine Vielzahl von Speicherobjekten hat, wobei jedes der Speicherobjekte mit der Datenbank kommuniziert und eine Kopie einer der Datenbankkomponenten speichert, auf die von mindestens einem der Vielzahl von Clients zugegriffen wird; und

ein oder mehrere Kommunikationsobjekte, die zu jeder der Client-Anwendungen gehören, wobei jedes der Kommunikationsobjekte mit einer zugehörigen Client-Anwendung und mit einem der Speicherobjekte kommuniziert;

wobei ein Speicherobjekt, auf das von zwei oder mehr Client-Anwendungen zugegriffen wird, mit zwei oder mehr Kommunikationsobjekten kommunikativ verbunden ist.

70. Prozeßsteuersystem nach Anspruch 69, dadurch gekennzeichnet, daß der Datenbankserver ferner eine Benachrichtigungsroutine enthält, die Veränderungen an einer der Datenbankkomponenten erfaßt und die automatisch jeden der Clients, die auf diese eine der Datenbankkomponenten zugreifen, von der Existenz der Veränderung an dieser einen der Datenbankkomponenten benachrichtigt.

71. Prozeßsteuersystem nach Anspruch 70, dadurch gekennzeichnet, daß die Benachrichtigungsroutine jeden der Clients, die auf die eine der Datenbankkomponenten zugreifen, über den Status der einen der Daten-

bankkomponenten nach der Veränderung benachrichtigt.

72. Prozeßsteuersystem nach Anspruch 70, dadurch gekennzeichnet, daß die Datenbank eine Konfigurationsdatenbank ist, wobei die Datenbankkomponenten Konfigurationskomponenten sind.

73. Prozeßsteuersystem nach Anspruch 70, dadurch gekennzeichnet, daß jedes der Kommunikationsobjekte so ausgelegt ist, daß es in das eine zugehörige Speicherobjekt schreiben und aus diesem lesen kann.

74. Prozeßsteuersystem nach Anspruch 70, dadurch gekennzeichnet, daß der Datenbankserver ferner einen Verriegelungsmanager enthält, der verhindert, daß eine erste der Kommunikationskomponenten auf eines der Speicherobjekte zugreift, wenn eine zweite der Kommunikationskomponenten auf das eine Speicherobjekt zugreift.

75. Prozeßsteuersystem nach Anspruch 69, dadurch gekennzeichnet, daß die Kommunikationsverbindung eines Satellitenkommunikationsverbindung ist.

76. Prozeßsteuersystem nach Anspruch 69, dadurch gekennzeichnet, daß die Kommunikationsverbindung eine zelluläre Kommunikationsverbindung ist.

77. Prozeßsteuersystem nach Anspruch 69, dadurch gekennzeichnet, daß die Kommunikationsverbindung eine Telefonleitungskommunikationsverbindung ist.

76. Prozeßsteuersystem nach Anspruch 69, dadurch gekennzeichnet, daß die Kommunikationsverbindung eine drahtlose Kommunikationsverbindung ist.

79. Prozeßsteuersystem nach Anspruch 69, dadurch gekennzeichnet, daß die Kommunikationsverbindung eine Wide Area Network Verbindung ist.

Hierzu 14 Seite(n) Zeichnungen

Nummer: DE 100 49 569 A1
Int. Cl.[7]: G 06 F 15/177
Offenlegungstag: 19. April 2001

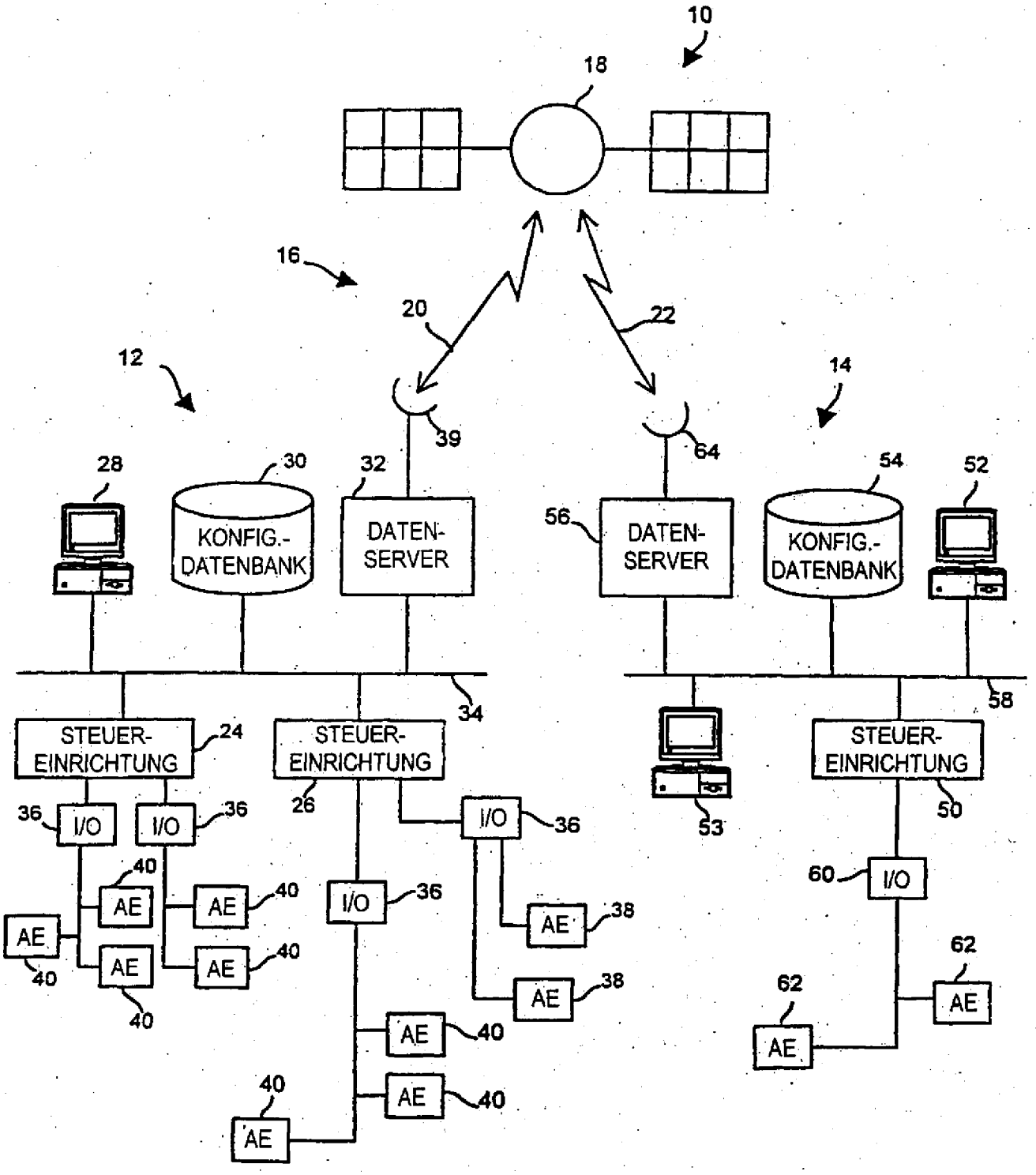


FIG. 1

Nummer: DE 100 49 569 A1
Int. Cl.[7]: G 06 F 15/177
Offenlegungstag: 19. April 2001

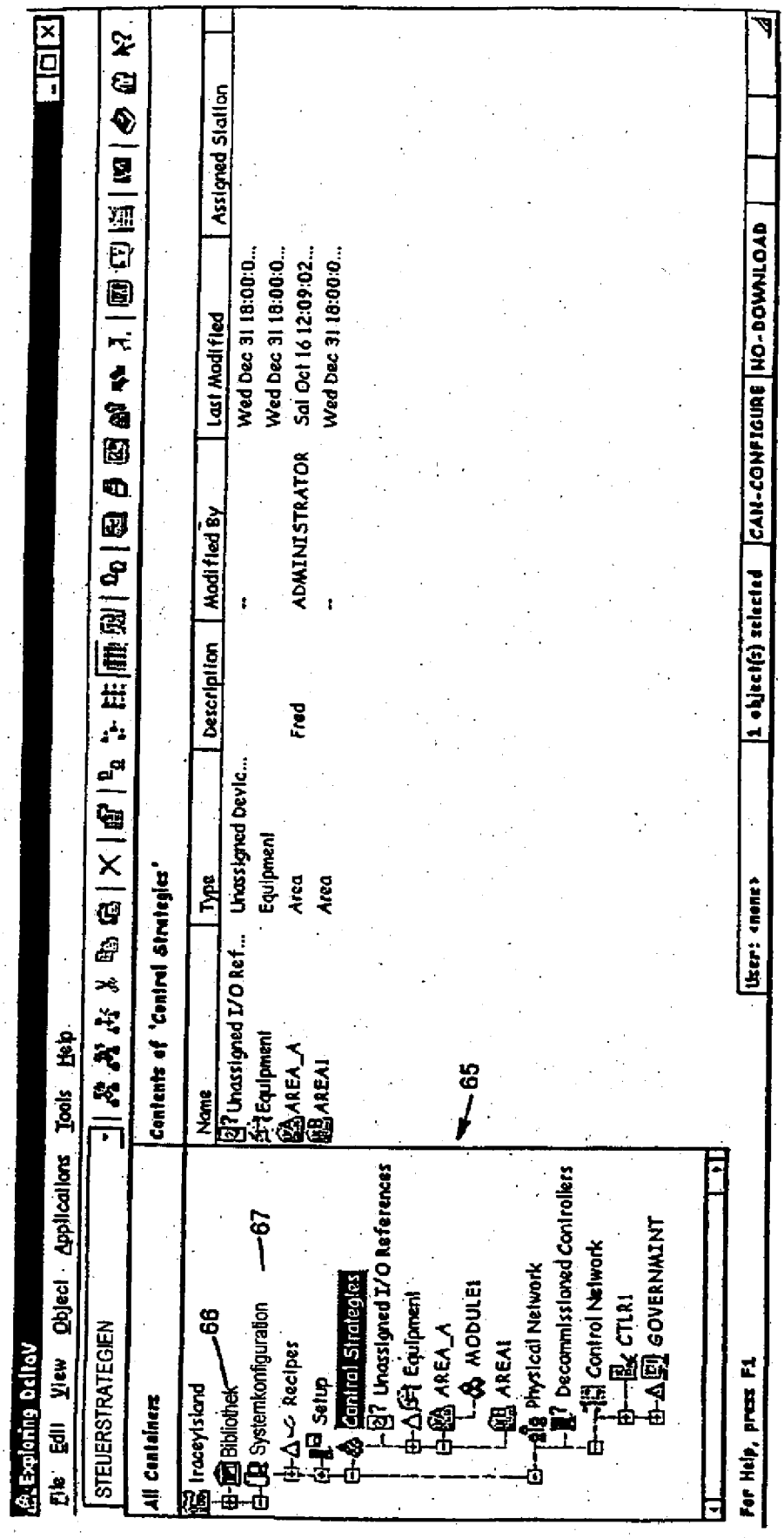


FIG. 2

Nummer: DE 100 49 569 A1
Int. Cl.[7]: G 06 F 15/177
Offenlegungstag: 19. April 2001

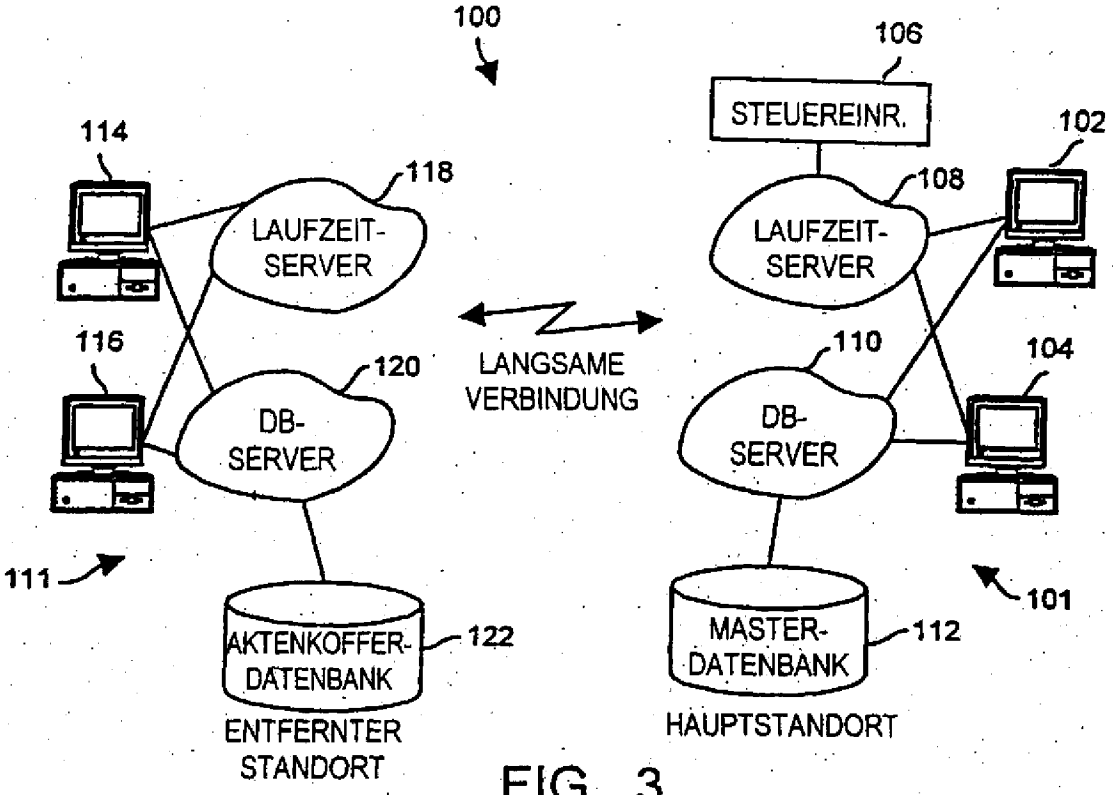


FIG. 3

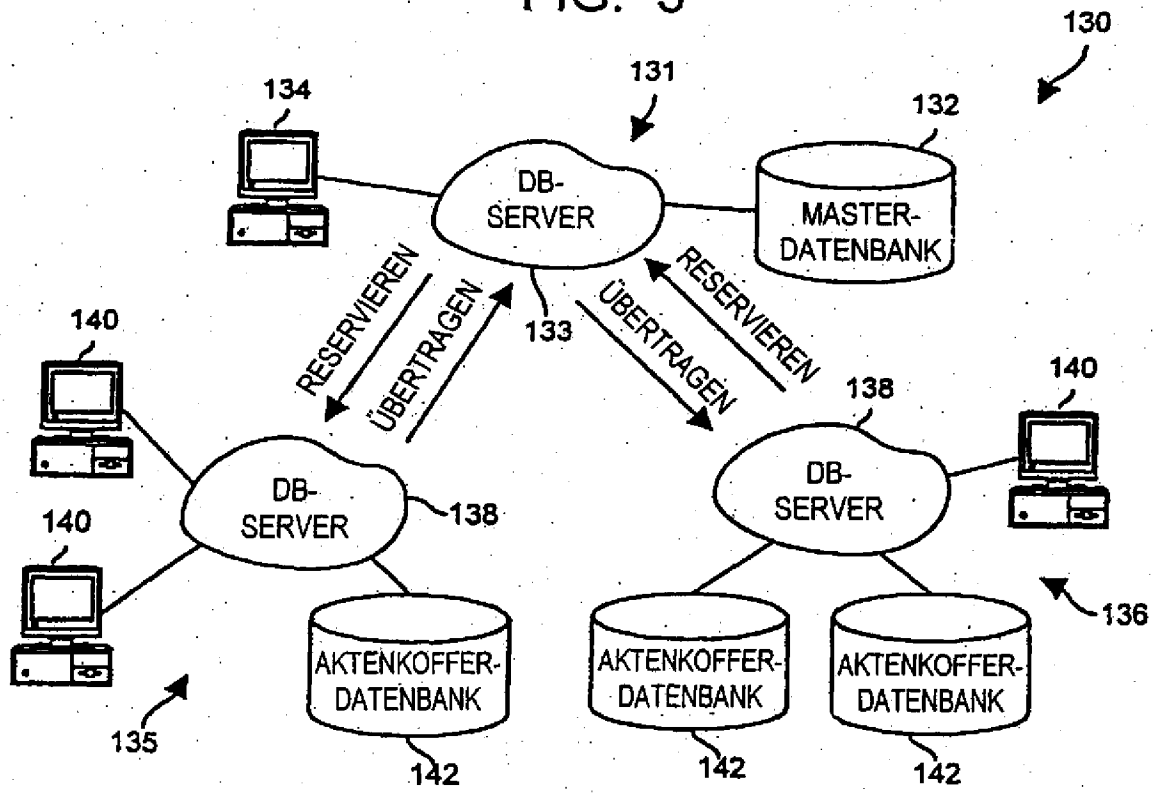


FIG. 4

Nummer: DE 100 49 569 A1
Int. Cl.[7]: G 06 F 15/177
Offenlegungstag: 19. April 2001

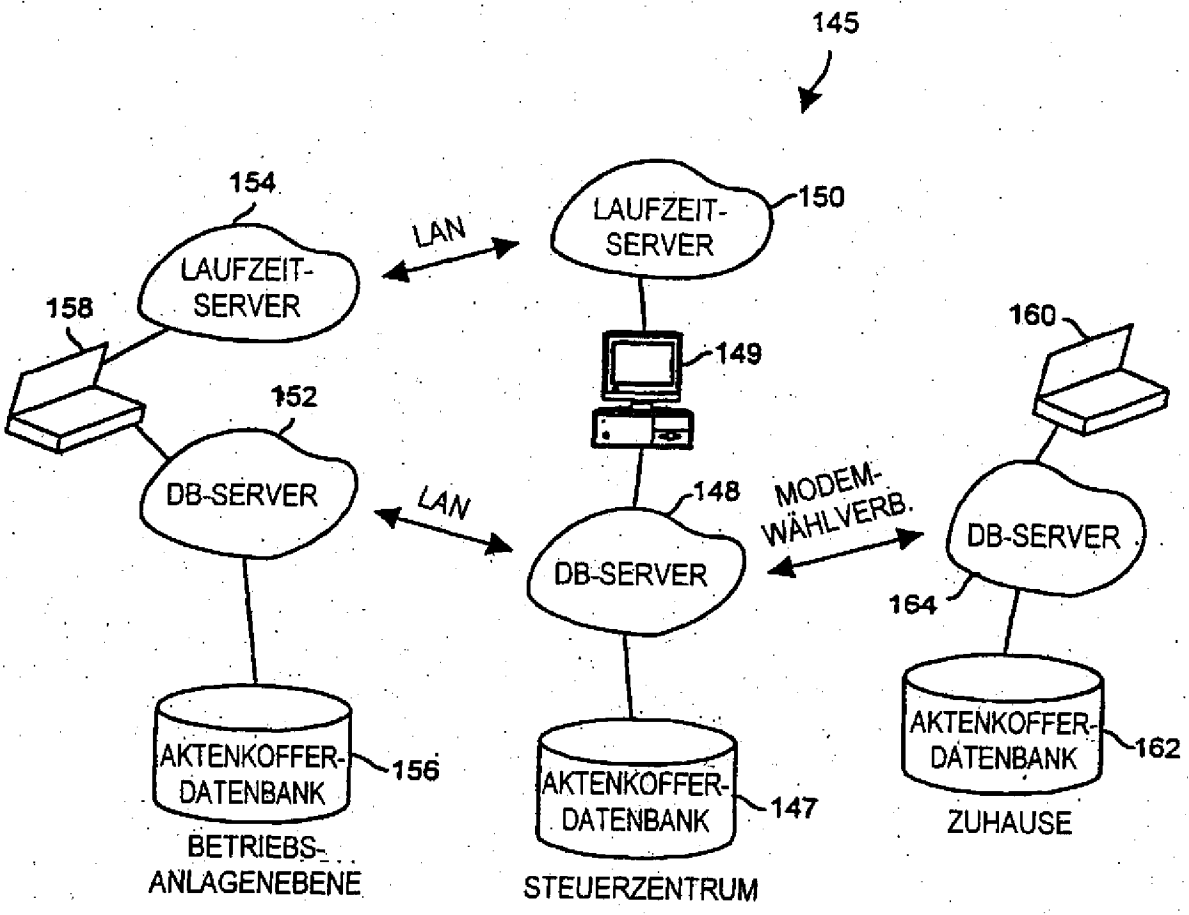


FIG. 5

Nummer: DE 100 49 569 A1
Int. Cl.[7]: G 06 F 15/177
Offenlegungstag: 19. April 2001

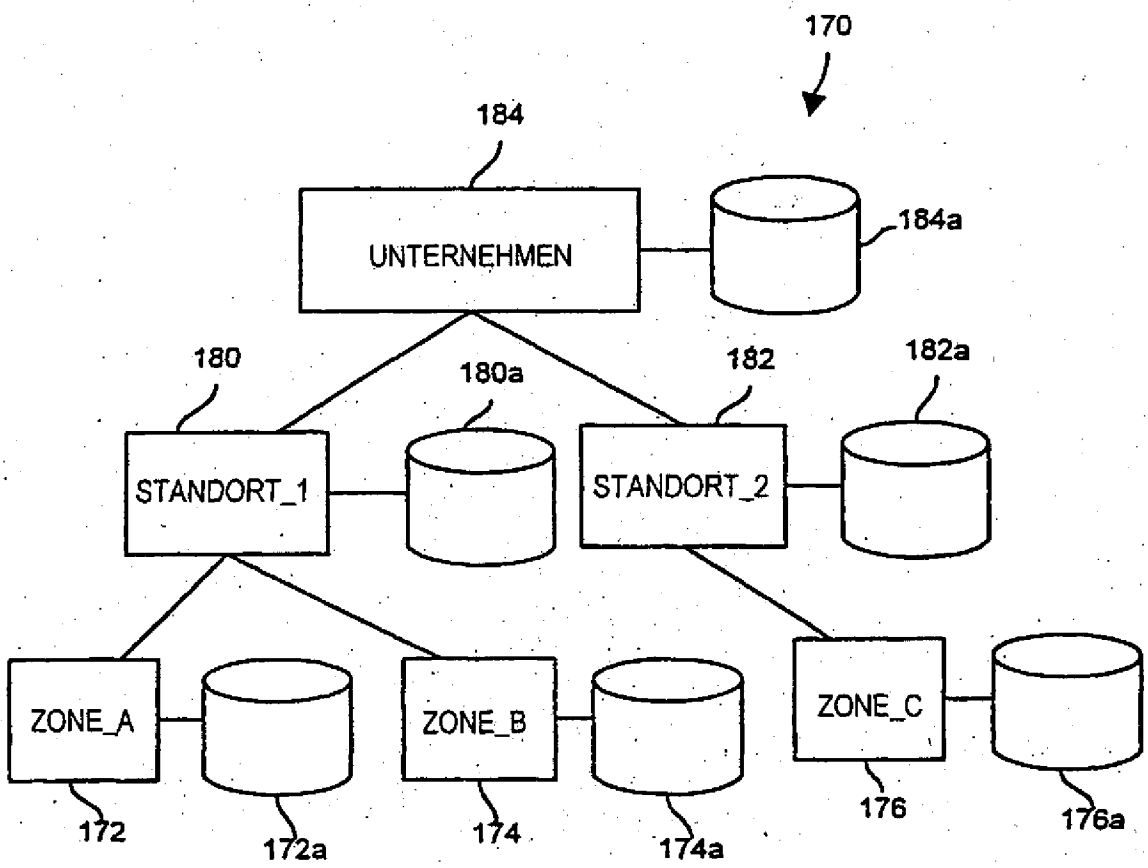


FIG. 6

Nummer: DE 100 49 569 A1
Int. Cl.[7]: G 06 F 15/177
Offenlegungstag: 19. April 2001

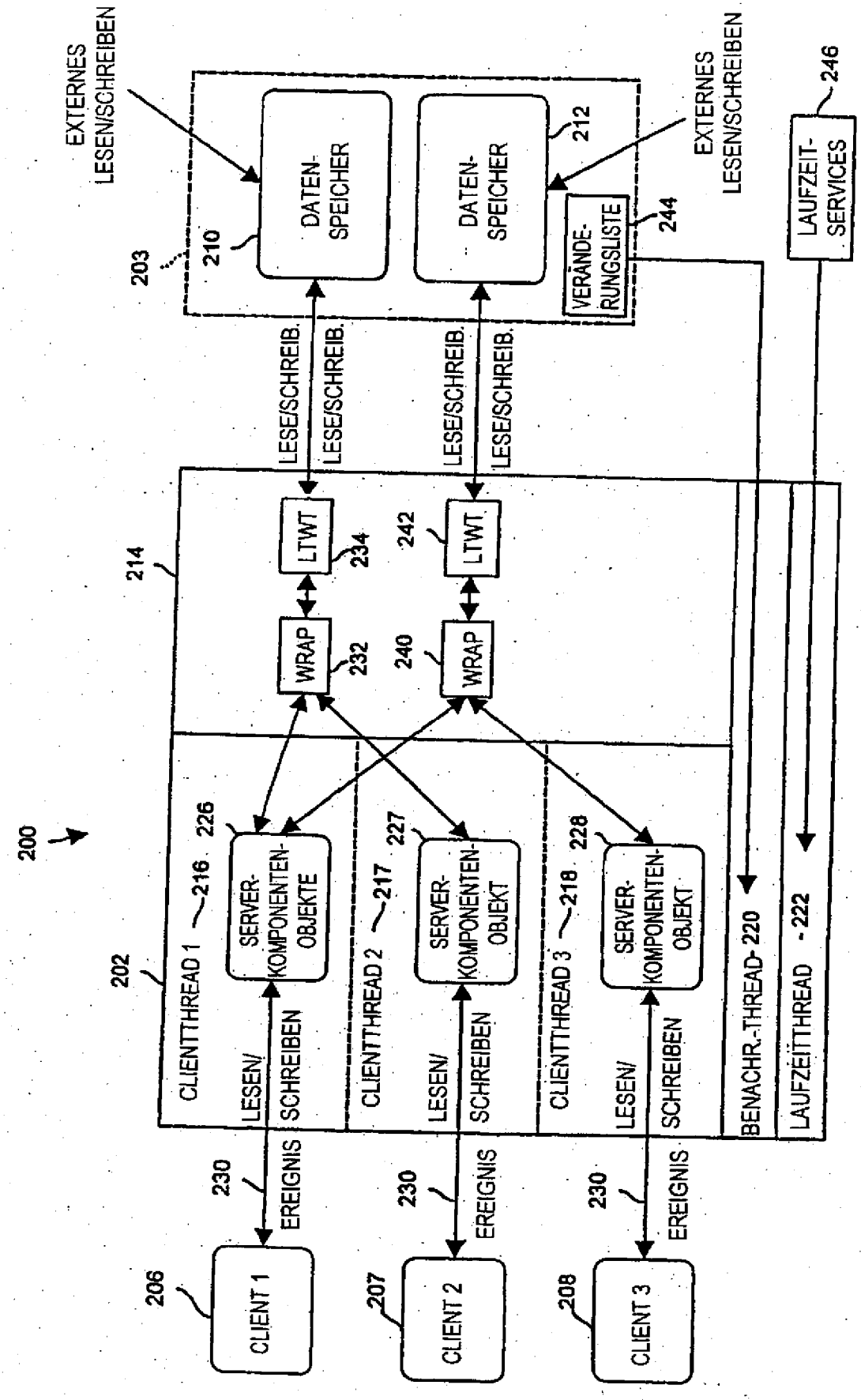


FIG. 7

Nummer: DE 100 49 569 A1
Int. Cl.[7]: G 06 F 15/177
Offenlegungstag: 19. April 2001

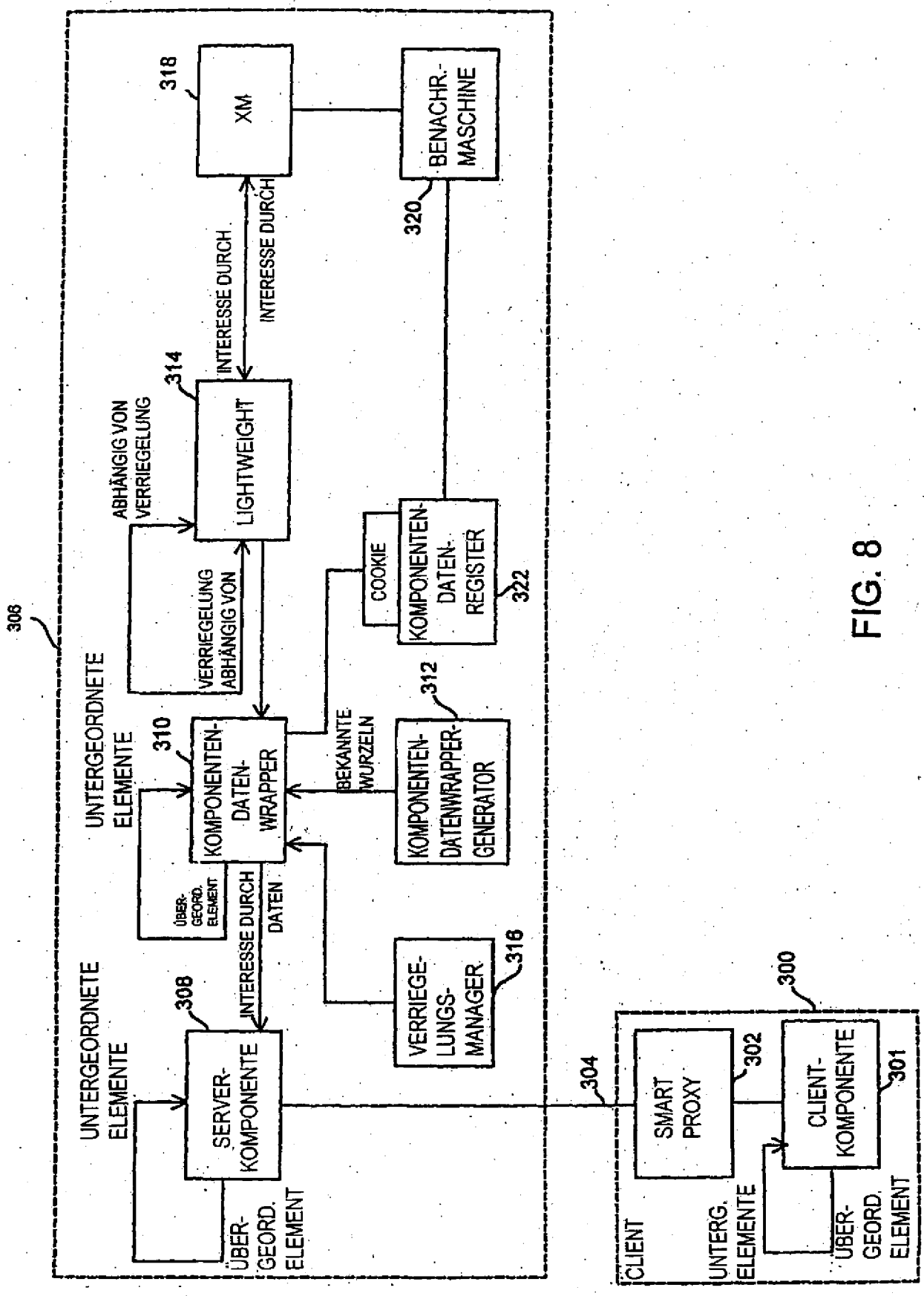


FIG. 8

Nummer: DE 100 49 569 A1
Int. Cl.[7]: G 06 F 15/177
Offenlegungstag: 19. April 2001

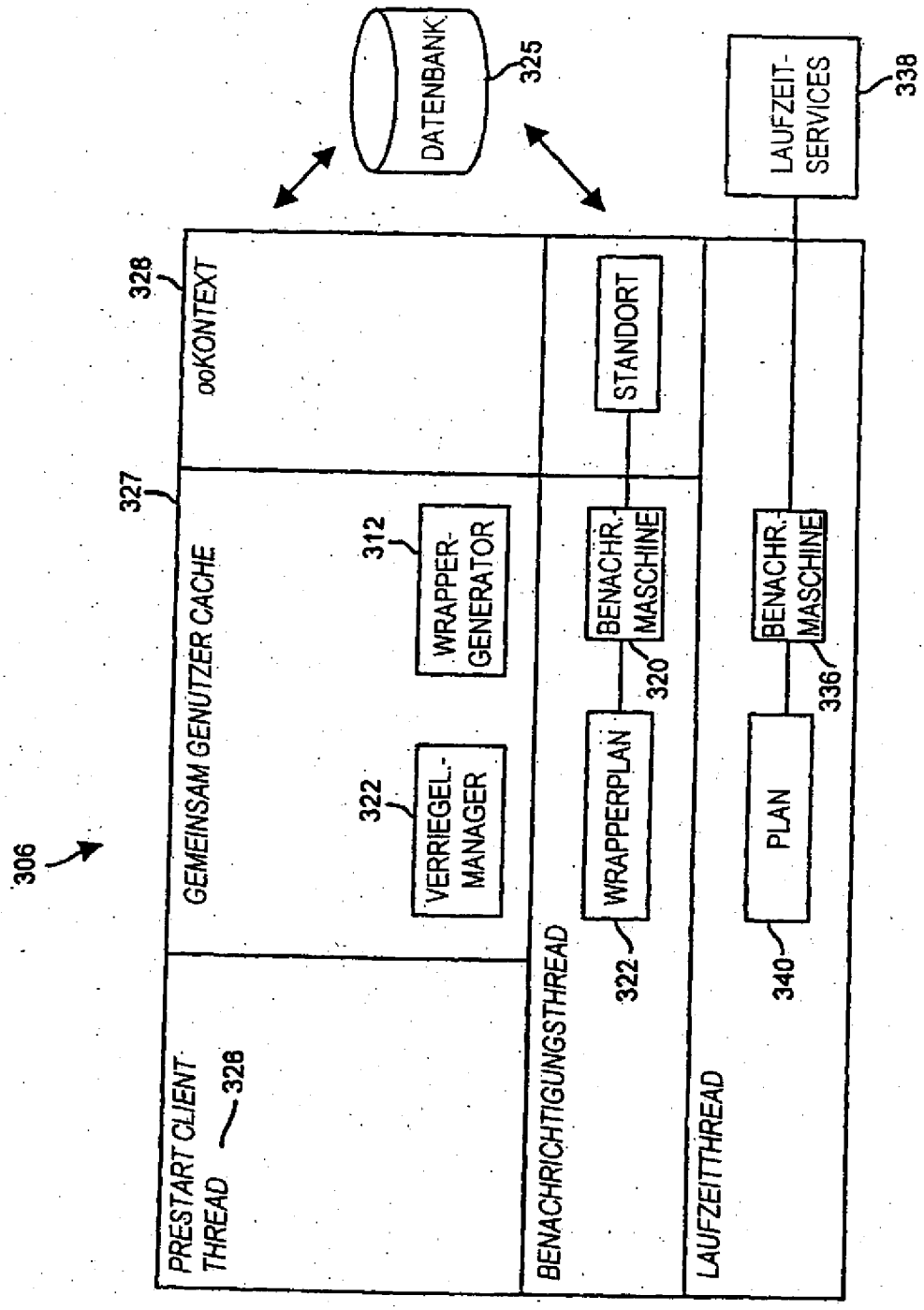


FIG. 9

Nummer: DE 100 49 569 A1
Int. Cl.[7]: G 06 F 15/177
Offenlegungstag: 19. April 2001

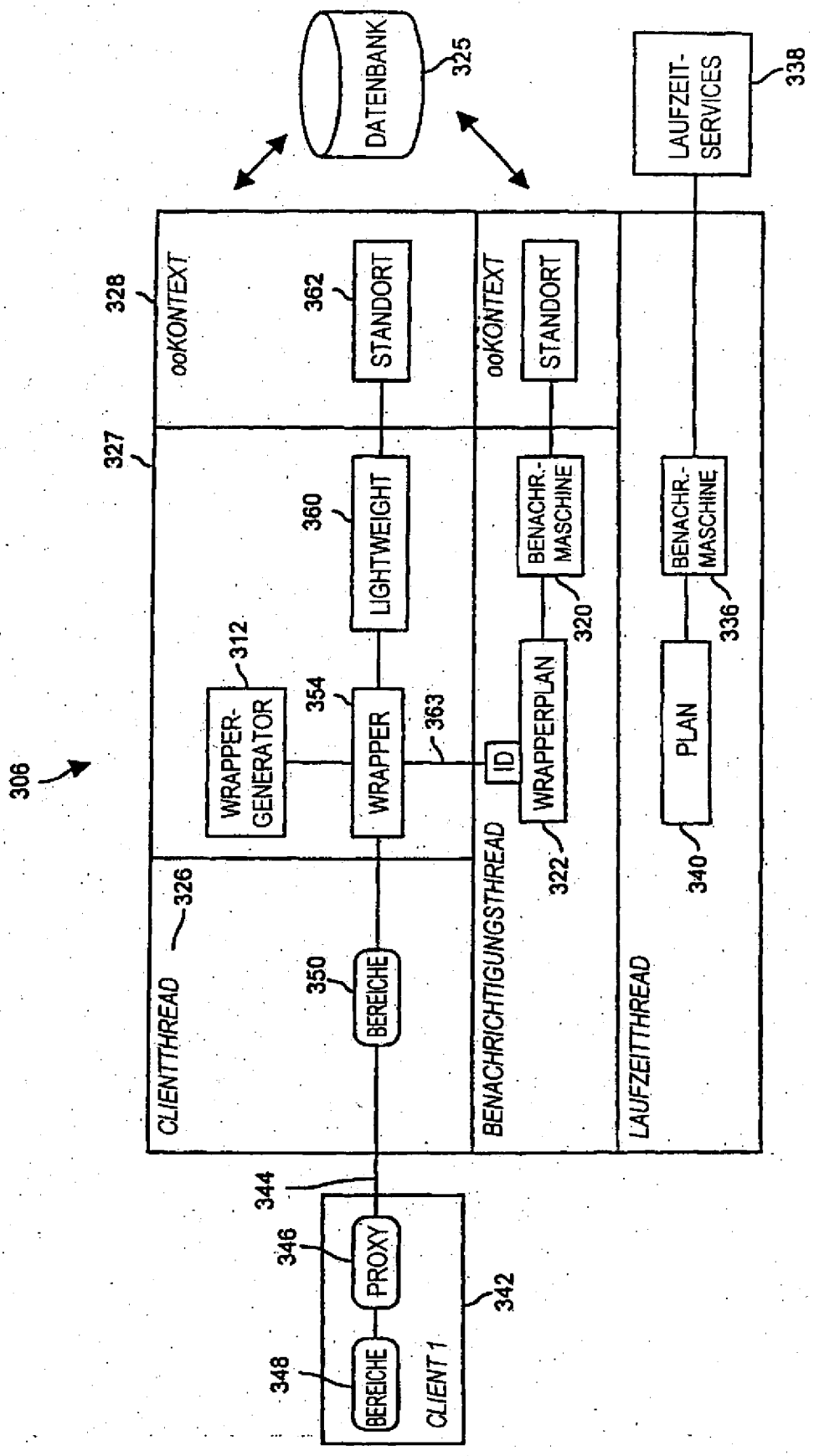


FIG. 10

Nummer: DE 100 49 569 A1
Int. Cl.[7]: G 06 F 15/177
Offenlegungstag: 19. April 2001

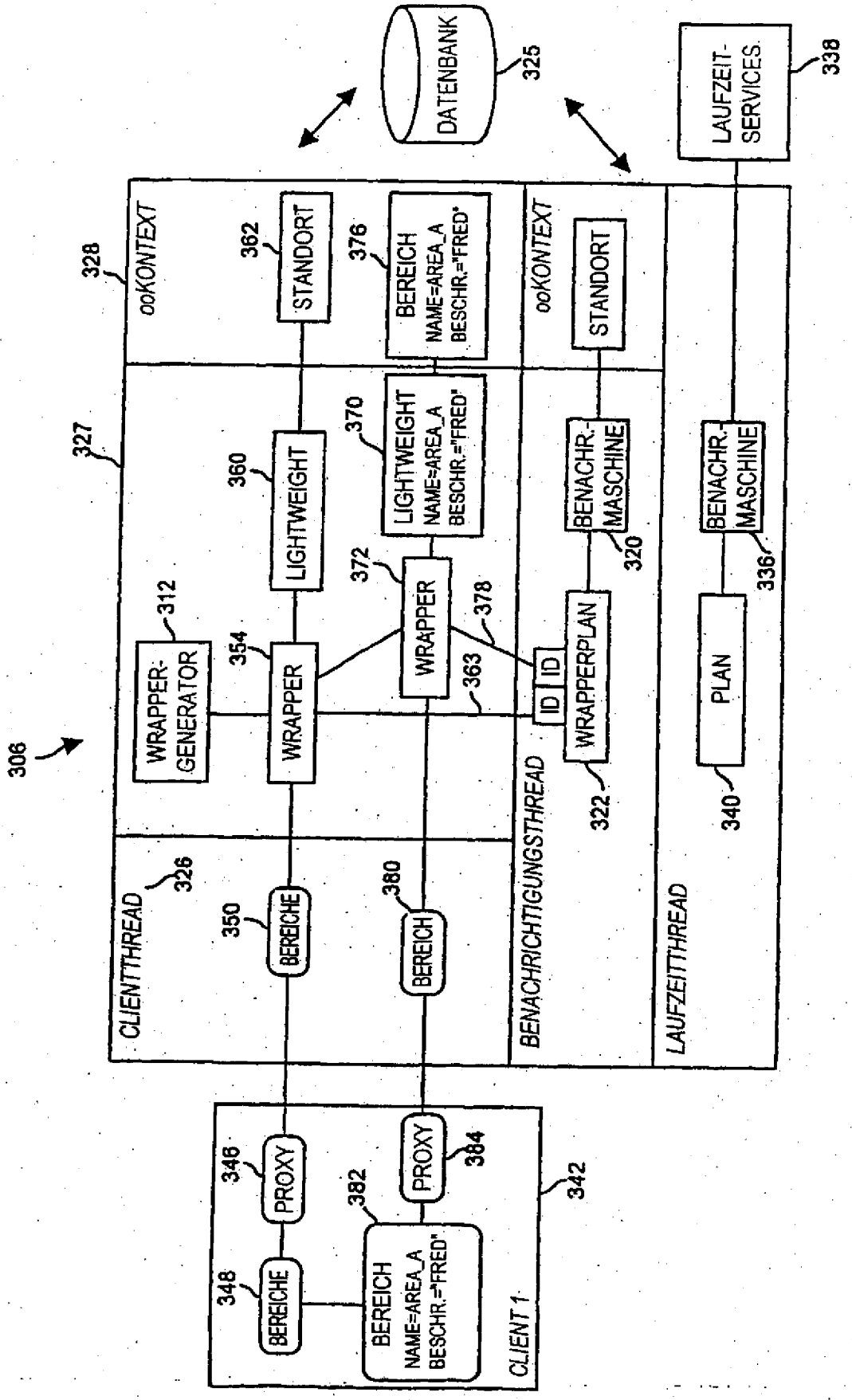


FIG. 11

Nummer: DE 100 49 569 A1
Int. Cl.[7]: G 06 F 15/177
Offenlegungstag: 19. April 2001

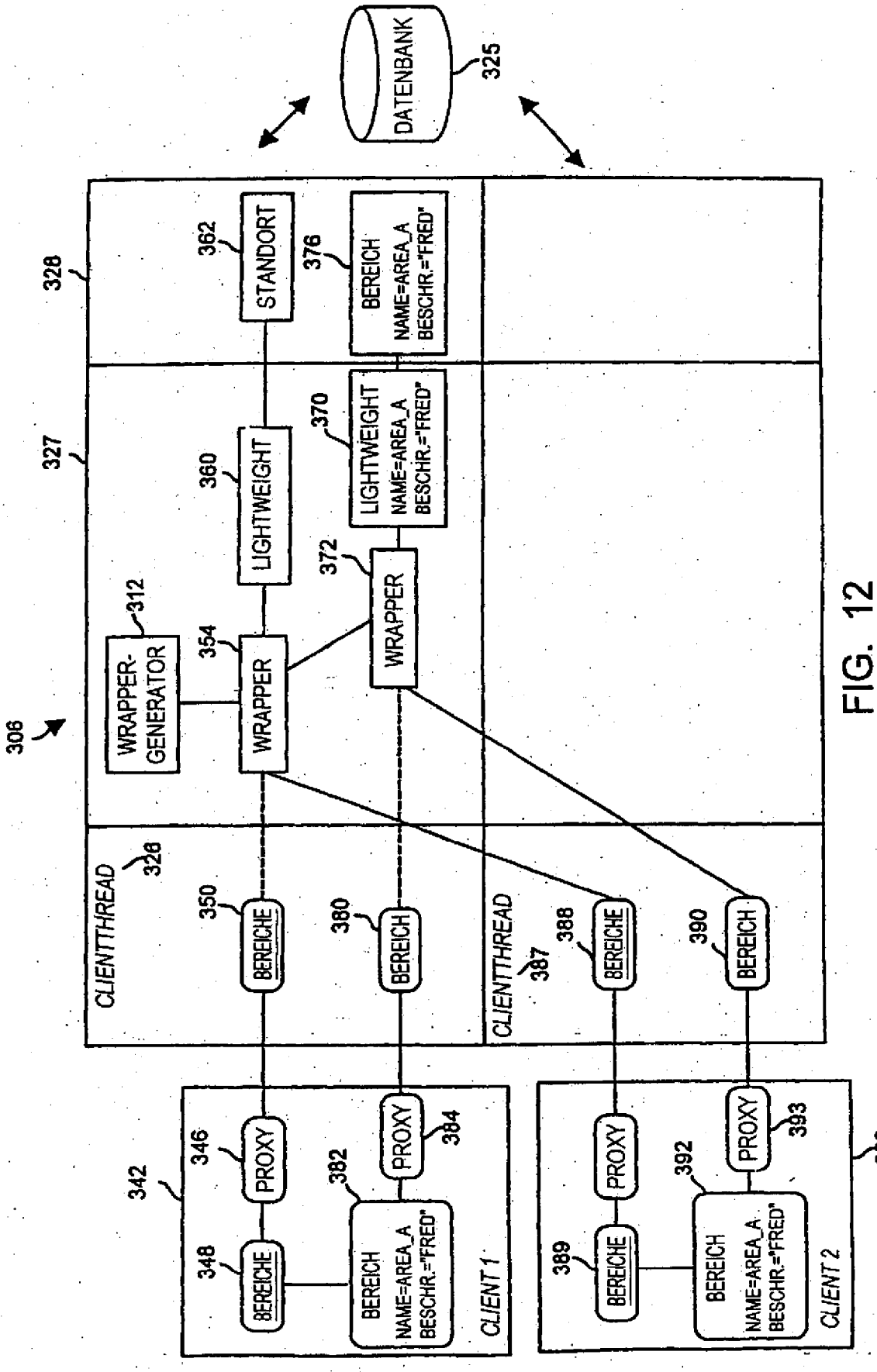


FIG. 12

Nummer: DE 100 49 569 A1
Int. Cl.[7]: G 06 F 15/177
Offenlegungstag: 19. April 2001

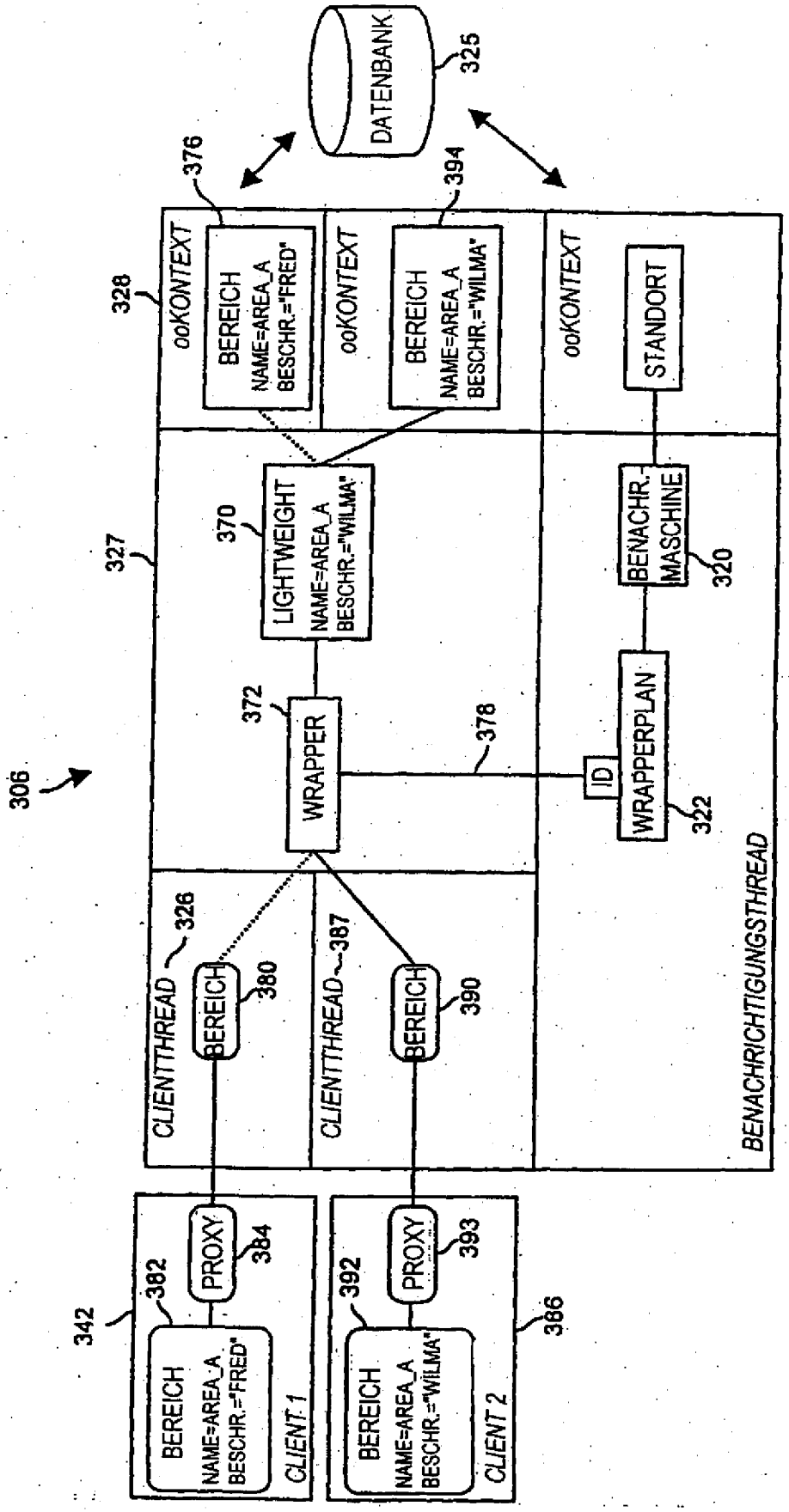


FIG. 13

Nummer: DE 100 49 569 A1
Int. Cl.[7]: G 06 F 15/177
Offenlegungstag: 19. April 2001

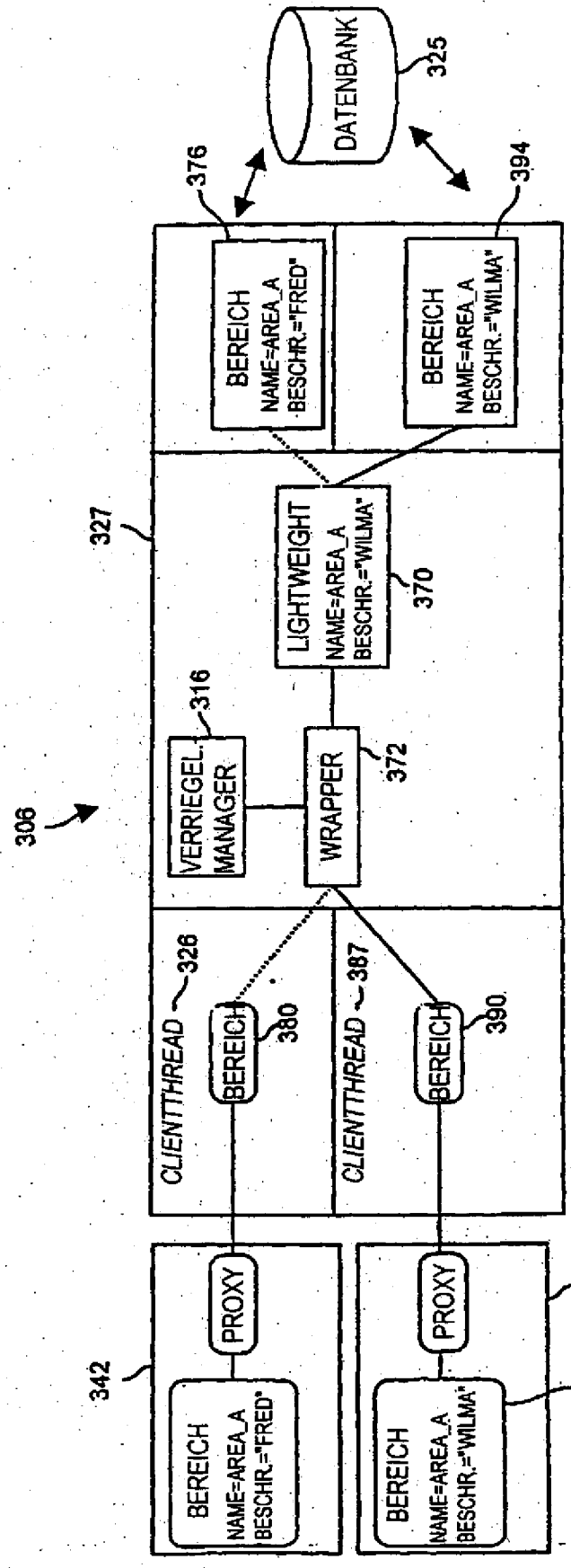


FIG. 14

Nummer: DE 100 49 569 A1
Int. Cl.[7]: G 06 F 15/177
Offenlegungstag: 19. April 2001

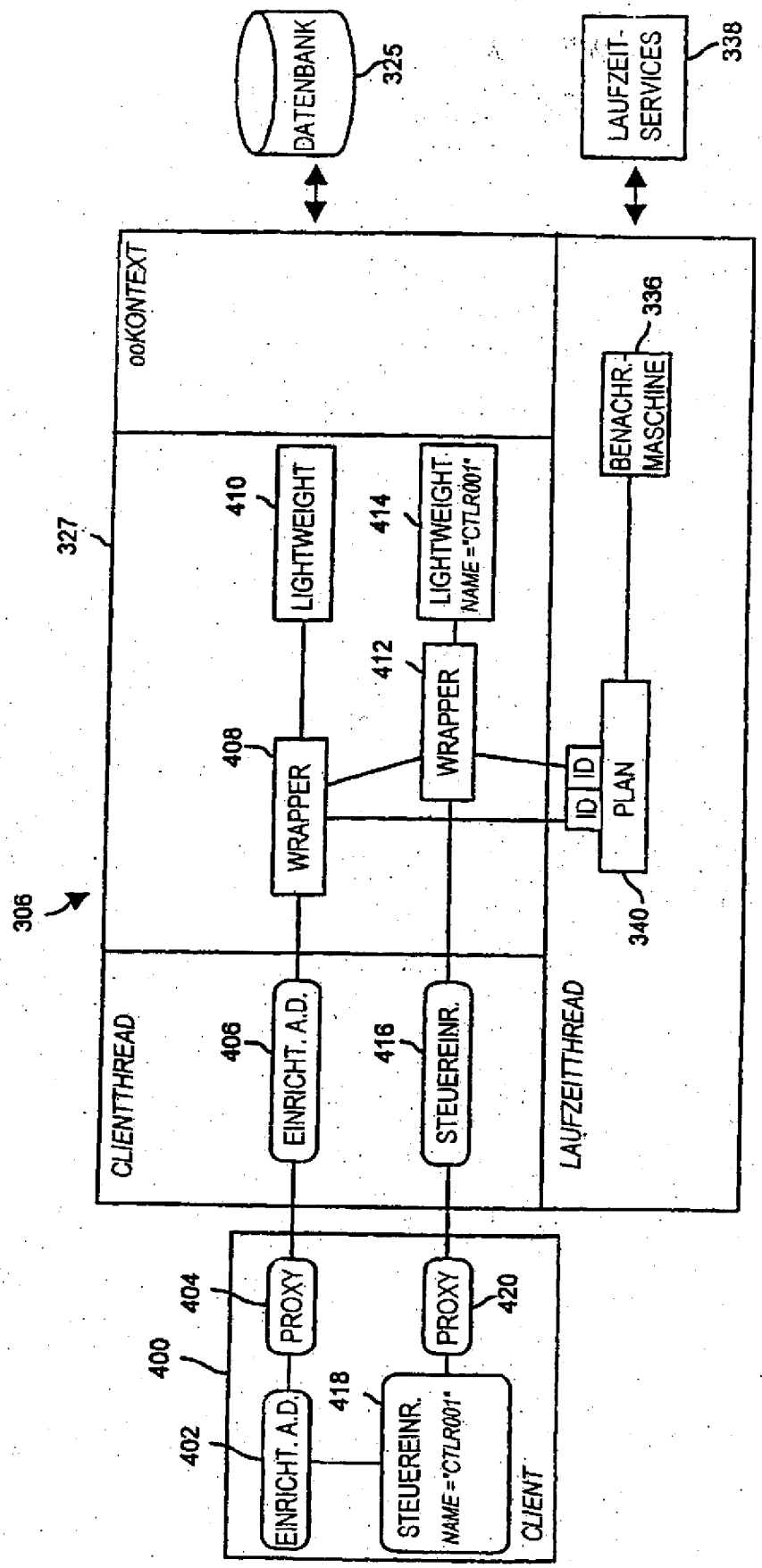


FIG. 15